Archipel : un *framework* objet pour une approche ubiquitaire de l'assistance cognitive

par

Jérémy Bauchet

thèse présentée au Département d'Informatique
en vue de l'obtention du grade de docteur ès sciences (Ph.D.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, Décembre 2008

Your file Votre référence
ISBN: 978-0-494-48528-6
Our file Notre référence
ISBN: 978-0-494-48528-6

Le 22 décembre 2008

le jury a accepté la thèse de M. Jeremy Bauchet dans sa version finale.

Membres du jury

Mme Hélène Pigot
Directrice
Département d'informatique

M. Sylvain Giroux
Codirecteur
Département d'informatique

M. Yves Lachapelle
Membre
Département de psychoéducation - Université du Québec à Trois-Rivières

M. Gilles Privat
Membre
Orange Labs Grenoble

M. François Pachet
Membre externe
Sony CSL-Paris

M. Froduald Kabanza
Président-rapporteur
Département d'informatique

Je me refuse simplement mais absolument à confondre la conscience de l'artiste, qui est une chose, avec sa sincérité, qui en est une autre [...]. Cette conscience exige que nous développions en nous le bon ouvrier. Mon objectif est donc la perfection technique. Je puis y tendre sans cesse, puisque je suis assuré de ne jamais l'atteindre. L'important est d'en approcher toujours davantage. L'art, sans doute, a d'autres effets, mais l'artiste, à mon gré, ne doit pas avoir d'autre but.

Maurice Ravel

À mes parents ...

Sommaire

La cognition humaine est une mécanique fascinante. Mais c'est aussi un ensemble de processus d'une grande fragilité, que le vieillissement pathologique, les troubles acquis ou développementaux viennent endommager, parfois détruire.

C'est au jour le jour, dans nos activités de la vie quotidienne, indispensables à notre bien être physique et moral, que notre cognition est mise à l'épreuve. Pour que celle-ci soit réussie par les personnes atteintes de troubles cognitifs, une aide s'avère souvent nécessaire. Sans cette aide, on ne parle pas d'échec mais de situation de handicap. Pour les personnes âgées présentant une démence de type Alzheimer, il devient alors difficile d'envisager vieillir au domicile, même si le stade de la maladie n'est pas trop avancé. Or ces populations sont en constante augmentation et l'accueil institutionnel ne suit pas. Pour les jeunes adultes touchés par une déficience intellectuelle, c'est l'accession à une vie autonome qui est remise en cause. Peut on alors assurer leur épanouissement ?

Que le problème soit envisagé sur un plan humain ou économique, nos sociétés doivent trouver des solutions pour favoriser l'autonomie fonctionnelle des personnes souffrant de troubles cognitifs. Parmi les alternatives possibles, la recherche technologique offre déjà des éléments de réponse pertinents. Ce projet de doctorat approfondit cette voie en abordant l'assistance cognitive à la réalisation d'activités domiciliaires complexes.

Archipel est le nom du prototype développé au cours de cette recherche. Ce *framework* offre un cadre pour la mise en place d'applications à forte sensibilité au contexte d'assistance cognitive. Les problèmes abordés concernent la représentation des connaissances, avec une approche par objets alliant la théorie des architectures à métaclasses complètes à la pratique en XML et Java, et offrant grâce à son protocole à métaobjets simple l'opportunité d'une représentation adaptable et adaptée. Le monitorage ou suivi de la réalisation d'activités est le second problème abordé. Il vise à s'assurer que les actions réalisées par la personne suivent une démarche appropriée. Une mise en œuvre basée sur le système conseiller EpiTalk est présentée. La prise en compte des erreurs réalisées

et la génération d'actes d'assistance adaptés est la troisième facette de ce *framework*. Enfin, les interactions homme-machine, permettant perception des actions et diffusion de l'assistance dans un esprit ubiquitaire constituent le dernier point de ce projet. Le bien-fondé de cette approche est justifié par le déploiement d'Archipel dans l'appartement laboratoire DOMUS et la réalisation d'une expérimentation du système par douze adultes présentant une déficience intellectuelle.

Remerciements

Je tiens naturellement à adresser mes premiers remerciements à Hélène Pigot et à Sylvain Giroux, qui m'ont encadré pour ce travail de thèse. Il y a quelques années maintenant, Hélène me parla d'une jeune équipe qui ambitionnait d'utiliser la technologie à des fins d'assistance cognitive. J'ignorais tout de cette thématique, mais le concept me paraissait des plus séduisants. Après un avant-goût en maîtrise, cette thèse m'a véritablement permis d'apporter ma pierre à ce noble édifice. Et si je leur suis reconnaissant de m'avoir fourni un cadre idéal pour cette réalisation, je tiens avant tout à les remercier pour la confiance qu'ils m'ont accordé tout au long de ces trois années. J'ai ainsi pu laisser s'exprimer mon intuition, tout en restant dans le droit chemin grâce à des échanges ouverts et constructifs.

Mes remerciements vont aussi à France Telecom pour son soutien financier, et plus largement à l'équipe de Gilles Privat. Merci à Jean-Paul Viboud pour son accueil en terre grenobloise.

Gilles m'a fait l'honneur de participer à mon jury de thèse. Je le remercie aussi à ce titre, ainsi que Froduald Kabanza qui en accepta la présidence. J'exprime de plus ma profonde gratitude à François Pachet, qui fit le déplacement depuis Paris pour prendre part à la soutenance. L'exploration d'une partie de ses travaux fut pour moi d'un très grand intérêt, je fus dès lors très honoré de son acceptation et de sa présence.

Yves Lachapelle a lui aussi accepté de prendre part à l'évaluation de ce travail. De par la nature interdisciplinaire de cette thèse, la présence d'un non-informaticien au sein du jury revêtait pour moi une grande importance. Qu'il en soit remercié.

Le point d'orgue de ce travail fut l'expérimentation menée par la clientèle du prototype. Dany Lussier-Desrochers a assuré la partie "humaine" de cette expérimentation. Il fut pour moi des plus agréables de travailler avec Dany et de voir ainsi prendre vie Archipel. Je lui adresse toute ma reconnaissance. Je remercie aussi grandement pour leur participation les bénéficiaires et l'équipe du Centre Notre-Dame de l'Enfant de Sher-

brooke.

Je remercie les membres du Laboratoire DOMUS au sein duquel a été réalisée cette thèse. Je salue tout particulièrement Francis Bouchard, notamment pour sa formidable prise en charge de nos soucis techniques quotidiens, ainsi que Nicolas Marcotte dont la compétence à toute épreuve a permis le déploiement des prototypes. Je remercie aussi Matthieu Castebrunet pour sa générosité, qui s'est encore manifestée lors de la soutenance. Merci aussi à Céline Descheneaux et Jean-Philippe Oudet pour les impressions. Enfin, un grand salut à quelques anciens du laboratoire, Alexandre Dion et David Pache, auquel j'associe sa compagne Fanny Guérin.

J'adresse enfin mes remerciements à Mounir Mokhtari pour son soutien lors des derniers mois de cette thèse.

Je ne terminerai pas ces remerciements sans un léger petit retour en arrière. Je garde à l'esprit un souvenir assez particulier, celui d'une rentrée, au lendemain de vacances de la Toussaint. Jamais je n'avais été aussi content de remettre les pieds dans une salle de cours. Et je ne peux que souhaiter ce ressentiment à tout étudiant. J'adresse ma sincère gratitude à Madame Mandille pour avoir su si bien guider mes premiers pas dans le monde de l'informatique, ainsi que mes profondes salutations à Thomas Quatrehomme, Vincent Leblanc et Serge Sfeir.

Cette époque, c'est aussi celle de la rue Desfriches ... Au numéro 29 y résidait une certaine Mademoiselle Laurence F. J'ai une petite idée d'où elle réside à ce jour, mais vous n'en saurez rien. Ah si, juste une indice, maintenant c'est Madame Laurence B. Elle et moi avons survécu à l'hiver québécois, et j'espère que de nombreuses autres belles aventures de toute nature nous attendent dans les prochaines années.

Enfin, je ne pourrais terminer ce chapitre sans citer mes parents. Remonter encore quelque peu dans le temps me permettrait de démontrer leur inconditionnel support. Qu'ils en soient profondément remerciés.

Table des matières

Liste des tableaux

Liste des tableaux

Liste des figures

Introduction

Les déficits cognitifs résultant de troubles acquis ou développementaux ont un impact significatif sur la réalisation des activités de la vie quotidienne (AVQ) des personnes concernées. Les maladies dégénératives, au premier rang desquelles se situent les démences de type Alzheimer, les traumatismes crâniens, les accidents vasculaires cérébraux ou encore les déficiences intellectuelles sont quelques exemples de pathologies neurologiques ou de troubles développementaux dont les conséquences sur le quotidien sont particulièrement importantes [52]. L'autonomie fonctionnelle de ces personnes est alors conditionnée à une aide extérieure, sans laquelle elles se retrouvent en situation de handicap. Elles n'ont en effet pas la capacité de mener à bien les AVQ essentielles à leur bien-être physique et moral [70].

En réadaptation, selon l'approche cognitive, trois voies sont traditionnellement envisagées pour apporter une réponse à cette situation de handicap [144]. La première est le *rétablissement de la conduite*, qui vise à rétablir le fonctionnement cognitif tel qu'il était avant l'atteinte cérébrale. Autrement dit, l'idée est de réapprendre ce qui a été perdu. Toutefois, cela impose que les ressources cognitives résiduelles de la personne le permettent. La seconde voie, dite de *réorganisation de la conduite*, a pour principe le recours à des fonctions présentes mais peu ou pas utilisées avant l'atteinte pour mettre en oeuvre un comportement proche de celui qui a été irrémédiablement perdu. Lorsque la capacité de rappel du nom des personnes connues est atteinte, une image mentale peut notamment servir de support au rappel. Enfin, la troisième voie de la réadaptation est la *modification de l'environnement* et l'utilisation de *mesures compensatoires*. Un exemple, pour une personne présentant une atteinte mnésique, est le balisage de trajets dans un environnement anciennement familier. Cette dernière voie, largement abordée en clinique, vise clairement à limiter l'impact des déficits cognitifs sur la vie quotidienne. L'objectif final n'est donc pas le déficit en tant que tel, mais bien le handicap qui en résulte.

Cette troisième approche laisse entrevoir bien des possibilités quant à sa mise en

oeuvre. Pour baliser un trajet, des papiers annotés et collés dans la maison sont des outils simples mais efficaces et parfois suffisants. De même, réorganiser une cuisine pour simplifier son utilisation est une modification de base mais sensée de l'environnement. Néanmoins, selon un courant bien présent dans nos sociétés modernes, la réadaptation a ouvert sa porte à un nouveau médium compensatoire : la technologie. On parle alors de *dispositif technologique d'assistance*, défini comme "tout objet, pièce d'équipement ou produit, qu'il soit commercial ou non, modifié ou personnalisé, qui est utilisé pour augmenter, maintenir ou améliorer les capacités fonctionnelles de la personne avec incapacités" [109]. Cette définition s'appliquant à toute forme d'incapacité, on parlera plus spécifiquement d'*orthèse cognitive* lorsque le trouble est de cette nature [94]. Rappelons qu'une orthèse vise à compenser une insuffisance alors qu'une prothèse implique le remplacement par un élément artificiel de la partie dysfonctionelle. Une orthèse cognitive est un outil d'*assistance cognitive*.

Actuellement, on retrouve en clinique deux grandes catégories d'orthèses cognitives [121]. La première est constituée de dispositifs technologiques fixes dédiés à l'assistance d'activités ciblées. Ces dispositifs, en voie de démocratisation, sont parfaitement illustrés par les piluliers électroniques. Pour certaines pathologies, la prise de médicaments est à la fois un incontournable et une problématique, car il faut respecter les heures de prise et la posologie. Or les ressources cognitives sous-jacentes peuvent être atteintes. La compensation va alors du rappel de la prise au ravalement de la dose par le pilulier si celle-ci n'a pas été effectuée. Connectée au réseau internet, l'orthèse peut aller jusqu'à aviser automatiquement un tiers de la situation[1]. L'autre grande catégorie est celle des orthèses cognitives mobiles. Par mobile, on entend les dispositifs du type assistant numérique personnel, où l'aspect compensatoire est fourni par une application logicielle. Cette application couvre par exemple le rappel d'activités ou la présentation des étapes de réalisation d'une AVQ. Si certaines de ces orthèses mobiles sont commercialisées[2], la plupart restent du domaine de la recherche.

Dans la mesure où les évaluations cliniques de ces outils montrent des résultats pertinents mais susceptibles d'amélioration[3], la question de l'avenir du couple assistance cognitive - technologie se pose : quelles orientations doivent être prises par les experts en

[1] http ://www.epill.com/md2.html

[2] Bien que l'on parle ici d'applications dédiées aux personnes avec troubles cognitifs, il est aussi fréquent que des logiciels grand public soient utilisés à des fins de réadaptation. Ces considérations seront présentées ultérieurement.

[3] Voir chapitre 2.

technologie pour mieux servir les personnes avec troubles cognitifs ? Car l'idée est bien là : la technologie doit fournir des *services* d'assistance cognitive. La personne assistée devient un *client* que la modification de l'environnement ou la mesure compensatoire doit satisfaire. La satisfaction se résume ici dans l'amélioration de son autonomie fonctionnelle résultant en la diminution - voire suppression - de la situation de handicap. Et que l'on s'intéresse à ces personnes, à leur vie, à leur place dans la société ou même à des considérations purement économiques, la demande est là.

Pour répondre à cette question, il est utile de revenir à la notion même d'activités de la vie quotidienne (AVQ), car celle-ci illustre en effet les considérations clés d'une vie autonome. Les AVQ trouvent leur origine dans les travaux de S. Katz, docteur en médecine, menés au Benjamin Rose Hospital de Cleveland au début des années 1960. Katz définit les AVQ comme les"activités que les individus réalisent de façon habituelle" [78]. Elles comprennent les activités d'hygiène, d'habillement, de déplacement, d'assouvissement des besoins naturels, de prise des repas et de contrôle des sphincters. Ce travail de définition avait pour objectif l'évaluation des compétences physiques des patients du Benjamin Rose Hospital, afin de fournir un guide objectif de l'évolution d'une maladie chronique et de l'étude du processus de vieillissement. Pour ce faire, Katz a construit une échelle permettant à un intervenant de noter, via une évaluation de la réussite de chacune des AVQ précitées, la compétence du patient. Or cette échelle se nomme *"Index of Independence in ADL*[4]*"*. Autrement dit, pour être indépendant, il faut être compétent. A cette notion d'AVQ, qui s'applique aux activités de base qu'une personne doit réaliser pour elle-même, s'ajoute celle d'Activités Instrumentales de la Vie Quotidienne (AIVQ) introduite par M. Lawton et E. Brody en 1969 [89]. Les AIVQ visent à étendre le champ des AVQ aux besoins domiciliaires. L'aspect "instrumental" porte ainsi sur des activités plus complexes en termes physiques et cognitifs : faire des achats, utiliser les transports en commun, cuisiner, faire son ménage ou la lessive, utiliser le téléphone, prendre des médicaments, gérer son budget. Ces activités, aussi qualifiées d'activités de la vie domestique [70], nécessitent en effet jugement et planification des tâches. Enfin, une troisième catégorie d'AVQ a été introduite dans la littérature plus récemment : les Activités Etendues de la Vie Quotidienne (AEVQ) [131]. Il s'agit là d'activités qui, à cause d'une modification de l'environnement, nécessitent de la part de la personne un effort d'adaptation et/ou d'apprentissage. Un exemple très simple est l'arrivée d'un téléphone programmable chez une personne âgée, dont l'usage est supposé bénéfique. Si la personne est capable d'appréhender l'utilisation

[4] *Activities of Daily Living*, soit l'équivalent anglo-saxon d'Activités de la Vie Quotidienne.

de cet outil et ainsi d'auto-améliorer son confort, son indépendance est optimale. Dans ce travail, le terme AVQ sera utilisé de façon générique, les activités plus particulièrement ciblées étant les AIVQ.

En somme, une orthèse cognitive doit venir combler le fossé, creusé par les troubles cognitifs du client, qui sépare son niveau de compétence de celui nécessaire pour la réalisation des AVQ essentielles à son indépendance. Dans cette optique, cette thèse vise à apporter des réponses ou pistes de réponse pour l'avenir de l'utilisation de la technologie à des fins d'assistance cognitive. Immense problématique, elle est abordée ici sous l'angle du développement technologique. Ainsi, ce mémoire présente Archipel, un prototype logiciel dont le *framework* et les mises en oeuvre s'attaquent à l'aide à la réalisation d'AVQ domiciliaires complexes, selon une approche ubiquitaire. Dans un premier temps, on s'intéresse aux objectifs compensatoires d'Archipel. Quatre troubles cognitifs catalogués comme particulièrement handicapant pour la réalisation d'AVQ, et communs à de nombreuses pathologies cognitives et troubles développementaux, sont introduits (chapitre 1). En les explicitant, on justifie le besoin d'assistance cognitive et on ouvre la porte à la question centrale de cette thèse : quel rapport peut-il exister entre l'assistance cognitive et la technologie ? En effet, la littérature ne fait écho d'aucun cadre fonctionnel permettant de voir sous un même angle la personne, ses besoins et l'orthèse qui tente d'y répondre. C'est ce que propose le chapitre 2, en introduisant une cadre pour l'étude et la conception des orthèses cognitives, construit autour des concepts de *contexte d'assistance cognitive* (CAC) et de *sensibilité au CAC*. Dans un second temps, ce cadre est utilisé comme support pour l'analyse des dispositifs existants. Etant donné le type d'activités que l'on souhaite aborder, à savoir les AVQ domiciliaires complexes, on peut alors conclure sur l'intérêt d'une approche d'inspiration ubiquitaire. Ce paradigme de l'informatique et sa possible application à l'assistance cognitive sont discutés. Ces premières considérations permettant de poser clairement les objectifs de ce travail, le chapitre 3 en propose une illustration à travers un scénario d'assistance cognitive. Il y est question de réaliser une recette de cuisine, exemple typique d'activité domiciliaire complexe. Ce chapitre se termine par diverses observations méthodologiques, offrant une conclusion à la partie introductive de ce mémoire.

Les chapitres suivants détaillent, en quatre temps, la mise en oeuvre du cadre fonctionnel réalisée pour Archipel. On aborde tout d'abord, dans le chapitre 4, la représentation des connaissances nécessaires à l'assistance cognitive, qui est en fait celle des différentes facettes du *contexte d'assistance cognitive*. Un langage de représentation par objets est

alors proposé, ouvrant la voie à une utilisation de la connaissance décrite à des fins de manipulation du CAC et de raisonnement. Au chapitre 5, c'est toute la problématique des interactions homme-machine (IHM) qui est discutée. Pour Archipel, la solution s'inscrit dans l'application du paradigme d'informatique ubiquitaire, avec une mise en oeuvre intimement liée au langage de représentation défini au chapitre 4. Le premier aspect d'IHM discuté est celui de la perception des actions réalisées par la personne assistée. Le second porte sur la communication machine-homme pour l'assistance. Le premier ouvre la voie au suivi de la réalisation de l'activité, à des fins de détection des erreurs commises, comme présenté au chapitre 6. La sémantique à donner à cette problématique, la représentation des activités et la qualification des erreurs détectées sont les principales questions abordées. Une technique particulière de suivi de la réalisation est aussi présentée. Le second aspect d'IHM est repris par le chapitre 7, qui porte sur l'assistance devant faire suite à la détection et qualification d'une erreur. La mise en oeuvre de cette assistance y est abordée, autour de la notion d'*acte d'assistance*.

Le point d'orgue de ce travail est son évaluation en environnement réel, tout particulièrement auprès de la clientèle ciblée par cette recherche. C'est l'objet du dernier chapitre, qui présente les déploiements réalisés d'Archipel et son expérimentation par douze adultes présentant une déficience intellectuelle. On peut alors conclure sur le bien-fondé de l'approche retenue tout en mettant en évidence ses limites, ouvrant ainsi la voie à de futures pistes de recherche et de développement.

Chapitre 1

Troubles cognitifs et autonomie fonctionnelle

L'autonomie est "l'ensemble des habiletés permettant à une personne de se gouverner par ses propres moyens, de s'administrer et de subvenir à ses besoins personnels" [21]. Dans son fonctionnement quotidien, toute personne est tributaire de son degré d'autonomie, dont dépend sa capacité à réaliser seule ses AVQ. Parmi les déficits cognitifs résultant de troubles acquis ou développementaux, une étude réalisée en 2005 auprès de 72 intervenants professionnels et aidants naturels a isolé quatre items dont l'impact sur la perte d'autonomie fonctionnelle est particulièrement significatif [125]. Sous l'angle des besoins des personnes avec troubles cognitifs, ces déficits représentent des cibles que les dispositifs compensatoires doivent pallier, afin que le couple personne-orthèse atteigne l'autonomie fonctionnelle[1]. Les troubles mnésiques, exécutifs de planification et d'initiation, ainsi que les troubles attentionnels sont présentés ici.

1.1 Systèmes de mémoire et troubles cognitifs mnésiques

Le cerveau humain est proactif. Des liens physiologiques y sont continuellement constitués afin que l'individu soit capable d'acquérir de nouvelles connaissances sur son environnement. C'est ce qu'on appelle le processus d'apprentissage. Pour que cette acquisition soit

[1]Aujourd'hui, la première source d'aide pour l'autonomie fonctionnelle des personnes avec troubles cognitifs est la présence d'un aidant naturel ou professionnel. Si la recherche menée ici est centrée sur la technologie, elle n'écarte absolument pas l'humain du cercle d'assistance, dont elle pourrait par contre soulager la charge.

effective, l'apprentissage est combiné à un autre processus : la mémoire. La mémoire est le processus par lequel les connaissances issues de l'apprentissage sont encodées, emmagasinées et récupérées.

La mémoire est un élément clé de la cognition humaine. Elle est en effet utilisée par toutes les fonctions cognitives, qui permettent à l'individu d'avoir un comportement adapté à une situation donnée. Dès lors, la prévalence de sa détérioration dans les pathologies cognitives et troubles développementaux en fait un enjeu privilégié de la recherche. Il existe bien entendu certaines spécificités et corrélations entre pathologie ou trouble et spécificité de l'atteinte mnésique, mais ces considérations dépassent largement le cadre de ce mémoire. On se bornera ici à l'introduction des concepts sans faire état de leur origine neuroanatomique ou pathologique. Le lecteur intéressé pourra se reporter à [23] pour une présentation approfondie de la question.

La neuropsychologie, discipline en charge de l'évaluation et de la réadaptation des troubles de la mémoire, organise cette dernière selon différents systèmes. Ce typage des phénomènes mnésiques s'est imposé dans la mesure où les études neuropsychologiques ont montré qu'il existait au sein de la mémoire un certain nombre de composantes distinctes, se différenciant notamment par la nature de l'information stockée et par le temps de stockage de l'information. Il a de plus été démontré que ces différentes formes de mémoire avaient des origines anatomiques différentes, l'atteinte d'une zone du cerveau pouvant impliquer la détérioration d'un type de mémoire mais pas d'un autre [6]. Les concepts de mémoire à court terme versus mémoire à long terme, de mémoire épisodique, sémantique et procédurale, ainsi que de mémoire prospective sont présentés ici. A noter que bien que différentes, toutes ces composantes de la mémoire fonctionnent en étroite collaboration et sont nécessaires pour que le comportement de la personne puisse être adapté à toutes sortes de situations.

1.1.1 Mémoire à court terme versus mémoire à long terme

La première disctinction à considérer repose sur la capacité et le temps de stockage de l'information. On oppose ainsi mémoire à long terme (MLT) et mémoire à court terme (MCT). La première permet de stocker une quantité a priori non limitée d'informations pendant un laps de temps potentiellement grand, et ce d'autant plus que l'information aura été apprise et réapprise (i.e. consolidée). La seconde n'offre la rétention que de quelques informations pendant un court laps de temps.

Parmi les sous types de MCT, la mémoire de travail introduite par A. Baddeley se définit comme l'ensemble des processus permettant de maintenir active l'information nécessaire à l'exécution des activités cognitives courantes [7]. Elle peut être considérée comme la mémoire à court terme d'épisodes successifs gérés et organisés dans une mémoire active nécessaire à l'accomplissement d'une tâche. C'est ce type de mémoire qui, lors de la réalisation d'une tâche avec procédure, comme une recette de cuisine, permet de savoir à quelle étape nous en sommes. L'expression "mémoire vive" est aussi utilisée pour la qualifier.

1.1.2 Mémoire sémantique, épisodique et procédurale

La seconde distinction concerne un ensemble de sous-systèmes de la MLT. Proposées par E. Tulving, ces formes de mémoire se distinguent par le type d'information stockée et la conscience du rappel. On distingue la mémoire sémantique, épisodique et procédurale [149]. J. Bachevalier propose la description suivante des concepts introduits par Tulving. La mémoire épisodique "stocke les évènements personnels indexés dans leur contexte spatial, temporel et émotionnel". Elle se distingue de la mémoire sémantique qui a trait à l'ensemble des connaissances du monde, y compris le langage. La mémoire sémantique peut être considérée comme "l'accumulation d'épisodes identiques qui, répétés, finissent par former une connaissance détachée de son contexte : elle est générique". La mémoire procédurale, quant à elle, est essentiellement une mémoire sensori-motrice, perceptive, voire cognitive. Elle permet aussi le "transfert de procédures acquises dans un contexte donné à un autre contexte". C'est la mémoire des savoir-faire [6]. Ainsi, les souvenirs liés aux dernières vacances en Italie sont épisodiques. Savoir que l'origan est une épice et qu'elle accommode parfaitement les pâtes est une connaissance sémantique. Enfin, pouvoir sans support réaliser un plat de spaghettis à la carbonara dépend des acquis procéduraux.

1.1.3 Mémoire prospective

Introduite dernièrement, la mémoire prospective se réfère aux souvenirs d'intentions ou d'actes devant être réalisés dans le futur, comme par exemple le fait de devoir aller chercher du pain en sortant du travail. Elle se distingue de la mémoire rétrospective, associée pour sa part aux souvenirs des intentions, actes, et évènements qui se sont déroulés dans le passé.

La mémoire prospective semble faire appel à plusieurs composantes cognitives, afin de (a) planifier les actes ou désirs, c'est-à-dire savoir quoi faire, quand et où le faire, (b) maintenir cette information jusqu'à la réalisation des actes, (c) réactiver spontanément le souvenir d'actes intentionnels et le contexte dans lequel ils doivent être accomplis, et (d) réaliser et évaluer la production de l'acte [62].

1.2 Troubles cognitifs exécutifs de planification et d'initiation

On appelle fonctions exécutives les capacités qui permettent à une personne d'avoir un comportement indépendant, intentionnel et intéressé. Elles rassemblent tous les processus cognitifs de haut niveau mis en jeu pour réaliser une tâche nouvelle ou complexe [102]. C'est en effet lors de l'exécution d'une activité que les fonctions exécutives se manifestent, d'où leur nom. Ainsi, c'est au niveau du contrôle et de la régulation de l'activité que leur perturbation se fera sentir. La littérature répertorie un certain nombre de fonctions exécutives. Dans ce travail, on se focalisera sur les troubles de la planification et de l'initiation, dont la détérioration est plus particulièrement responsable des problèmes de réalisation des AVQ [125].

M. Lezak définit la planification comme "l'identification et l'organisation des étapes et éléments nécessaires à la réalisation d'une intention et à l'atteinte d'un but fixé préalablement. Ce concept de planification englobe la capacité à élaborer différentes stratégies, à les évaluer et à en choisir une, puis à organiser les idées de façon séquentielle et hiérarchique afin d'en établir une structure qui donnera la direction à suivre pour réaliser le plan" [92].

Les mécanismes de planification sont censés intervenir dans les situations nouvelles pour la personne. Dans les situations qui lui sont familières, car répétées maintes et maintes fois, le comportement de la personne devient plus automatique qu'autre chose. Il n'y a pas de réelle identification et organisation des étapes, mais la restitution d'un savoir-faire stocké en mémoire (procédurale). C'est par exemple la différence entre un musicien expérimenté et un débutant dans l'utilisation de leur instrument. Quoiqu'il en soit, un trouble de la planification ou un problème mnésique de restitution d'un savoir-faire normalement acquis va se traduire par la réalisation de séquences inadaptées d'étapes étant donné le but exprimé.

De leur côté, les problèmes d'initiation se manifestent en amont de la réalisation de l'activité. Ils se concrétisent par de larges périodes d'inactivités alors que la personne devrait légitimement procéder à la réalisation [108]. Concrètement, il n'est pas rare de voir des personnes souffrant de troubles de l'initiation rester des heures dans leur cuisine sans rien faire, et ce malgré la sensation de faim.

Embrassant la notion de fonctions exécutives, l'autodétermination est introduite par les auteurs pour couvrir aussi bien les concepts cognitifs sous-jacents à l'instauration d'un comportement intentionné que ceux permettant une introspection sur les situations rencontrées. L'autodétermination est la "capacité d'une personne à agir comme un agent de premier plan dans sa vie et à réaliser des choix et prendre des décisions concernant la qualité de sa vie, le tout sans influence ou interférence externe" [87]. Un comportement auto-déterminé fait ainsi référence à un comportement basé sur (1) la capacité à indiquer ses préférences, réaliser des choix et initier une action, (2) la capacité à discerner les éléments d'une situation grâce à son jugement personnel et à anticiper les conséquences possibles de ses actes, (3) les multiples dimensions de la perception du contrôle, incluant les aspect cognitifs, la personnalité et la motivation, et (4) une connaissance effective et éclairée de ses propres forces et limitations [156]. Cette problématique complexe est très présente chez les personnes présentant une déficience intellectuelle, clientèle qui sera mise en avant dans les résultats expérimentaux de cette recherche. L'autonomie fonctionnelle, objectif compensatoire de ce travail, est en effet une composante essentielle de l'autodétermination.

1.3 Troubles cognitifs attentionnels

Le cerveau dispose d'une capacité limitée de traitement de l'information sensorielle. Cela implique qu'une partie de l'information que le cerveau reçoit ne sera pas traitée. Dès lors, il semble impératif que le cerveau soit à même de faire un tri parmi ces informations, afin de prioriser le traitement des informations vitales et de n'ignorer que ce qui peut l'être. Cette charge est dévolue au système attentionnel. Celui-ci fonctionne comme un filtre complexe, chargé d'effectuer un tri parmi les stimuli sensoriels afin d'en optimiser le traitement.

La littérature du domaine admet généralement quatre états pour le système attentionnel, qui sont déterminés par leur coût cognitif [92]. Le premier est l'alerte ou état d'éveil. Cet état correspond au niveau de mobilisation énergétique minimal de l'organisme per-

mettant au système nerveux d'être réceptif de façon non-spécifique à toute information lui parvenant. Le second état est celui de l'attention soutenue. L'individu est alors à même d'orienter intentionnellement son intérêt vers une ou plusieurs sources d'informations et de maintenir cet intérêt pendant un certain temps. La vigilance nécessite un état d'attention soutenue. Le troisième se nomme attention sélective ou focalisée. Il permet de trier les informations disponibles dans le but de ne retenir et de ne traiter que celles qui sont pertinentes pour l'activité en cours en inhibant la réponse aux autres stimuli présents. Enfin, la quatrième est l'état d'attention divisée ou partagée. Il représente l'habilité requise pour partager une attention sélective entre deux ou plusieurs sources distinctes, tout en détectant des stimuli pouvant appartenir à différentes sources simultanément.

Lorsque la mécanique de filtre attentionnel est perturbée, la personne peut éprouver de la difficulté à focaliser son attention sur l'activité qu'elle tente de réaliser. N'importe quel stimulus, même minime, est alors bon pour venir perturber la réalisation et faire oublier à la personne ce qu'elle était en train de faire. Il y a en effet une relation directe entre conservation des informations en mémoire à court terme, et donc en mémoire de travail, et contrôle de l'attention. Ces troubles peuvent donc conduire à des situations dangereuses pour la personne, notamment si l'activité oubliée impliquait l'utilisation d'éléments dangereux (comme une cuisinière laissée allumée).

1.4 Impact des toubles mnésiques, exécutifs et attentionnels sur la réalisation des AVQ

Pour illustrer l'impact des troubles mnésiques, exécutifs et attentionnels sur la réalisation des AVQ, imaginons le quotidien de M., un jeune homme dans la vingtaine, victime d'un accident de la route ayant entraîné un traumatisme crânien sévère 15 mois auparavant[2].

Après plusieurs mois d'hospitalisation, en milieu hospitalier puis dans un centre de réadaptation, M. est rentré chez ses parents où il participe à différentes activités domestiques. Il continue son cheminement de réadaptation en se présentant plusieurs fois par semaine au centre. Lors de cette dernière phase, un stage dans une petite entreprise en lien avec ses centres d'intérêt lui a été proposé, à raison de deux demi-journées par semaine.

On note chez M. une atteinte massive de la mémoire, ce qui l'invalide particulièrement

[2]Ce cas s'inspire largement d'échanges réalisés avec les intervenants d'un centre de réadaptation.

dans ses activités quotidiennes. Tout d'abord, il oublie régulièrement ses rendez-vous au centre, dont les horaires changent d'une semaine à l'autre. De même, il a de la difficulté à penser à sortir les poubelles le dimanche soir, charge domestique que sa mère lui a attribuée. Pour pallier ces troubles prospectifs, l'ergothérapeute travaille avec M. à l'apprentissage de l'utilisation d'une mesure compensatoire, en l'occurrence un agenda papier-crayon. Mais à l'heure actuelle, c'est surtout la vigilance de sa mère qui compense ses oublis. De même, lorsqu'il est prévu que M. soit seul au domicile pour un repas, c'est à nouveau sa mère qui lui prépare le contenu, M. n'ayant qu'à réchauffer ce dernier au micro-onde. Réaliser une recette seul nécessite de trop grandes ressources en terme de planification pour M. Une première solution pourrait résider dans l'achat d'un livre de recette adapté, systématisant les étapes de réalisation de l'activité et utilisant des supports imagés pour illustrer les étapes délicates. Les livres traditionnels, même s'ils contiennent des recettes simples, offrent trop d'alternatives à la réalisation et utilisent un lexique parfois ambigu.

Pour son travail de stage, qui consiste à nettoyer des voitures, M. utilise une fiche sur laquelle les étapes de son travail sont précisées. Ce support procédural lui est indispensable, car il oublie fréquemment de nettoyer l'intérieur des vitres ou de vider le cendrier, ce qui n'est pas compatible avec les attentes de son employeur et des clients. De même, il peut utiliser cette fiche pour noter où il en est dans le nettoyage. Cela lui permet de compenser une mémoire de travail qui est facilement perturbable, et ce d'autant plus que M. se fatigue très vite et que sa capacité à garder une attention soutenue sur sa tâche s'en ressent. Mais ceci arrive fréquemment pour une personne dont une partie du cerveau seulement doit gérer l'ensemble de la machinerie humaine.

Enfin, M. et ses amis aiment faire du sport et aller au cinéma. Malheureusement, les souvenirs de M. sont défaillants, ce qui limite son interaction sociale. Difficile en effet de reparler de la sortie au cinéma réalisée la fin de semaine précédente lorsqu'il n'est pas possible de replacer dans son contexte temporel et spatial cette activité dont tout le monde parle. C'est pourquoi l'ergothérapeute incite M. à marquer, sur son agenda, les réalisations de sa journée. L'objectif est que M. puisse se constituer un vécu.

Cette introduction au fonctionnement cognitif humain doit permettre une meilleure compréhension des enjeux du développement d'orthèses cognitives. La complexité des caractéristiques et besoins des personnes avec troubles cognitifs induit celle de la recherche du domaine. Elle est aussi la source de sa richesse. Dès lors, afin de mieux appréhender les

problématiques en jeu, il est intéressant de tenter de formaliser le rapport entre assistance cognitive et technologie. Cette démarche, qui fait défaut dans la littérature actuelle, est proposée dans le chapitre 2. Y sont définis le *contexte d'assistance cognitive* et la *sensibilité au contexte d'assistance cognitive*, ce deuxième aspect faisant rentrer la technologie au sein du processus d'assistance.

Chapitre 2

Technologie et assistance cognitive

Les nombreuses revues de littérature présentées ces dernières années témoignent de l'intérêt des chercheurs et des cliniciens pour l'utilisation de la technologie à des fins d'assistance cognitive [69, 94, 121, 134]. Les nouveaux besoins exprimés par nos sociétés, découlant notamment du vieillissement de la population, tel que souligné par M. Pollack [126], expliquent en partie cet engouement. Celui-ci est aujourd'hui justifié par la présence effective de résultats, comme le rapportent les revues citées.

Dans ce travail, la technologie est envisagée comme un outil de *compensation*, la littérature faisant aussi écho de recherches du côté de la *restauration* et de la *prévention* des troubles cognitifs. Ainsi, l'expression *orthèse cognitive* désignera un dispositif à base de technologie utilisé à des fins d'assistance pour la compensation de troubles cognitifs lors de la réalisation d'AVQ. Cette définition reprend en partie les critères définis par E. Cole [35, 36], en omettant les capacités de personnalisation de l'orthèse, posés par cet auteur comme indispensables pour l'effectivité de l'assistance, et l'utilisation des données recueillies à des fins d'analyse puis d'amélioration du dispositif.

R. Baecker a proposé plus récemment un ensemble de critères de classification des dispositifs technologiques pour la cognition [8, 9]. L'identification des processus cognitifs ciblés, des spécificités de la clientèle, de la place des aidants dans le processus, du design appliqué à la conception et enfin du type de technologie doivent permettre, selon Baecker, une meilleure compréhension et comparaison des projets existant ainsi que l'identification d'approches nouvelles ou négligées[1].

L'approche de Baecker est intéressante : un cadre conceptuel manque en effet aux

[1]Baecker ne limitant pas sa classification aux dispositifs de compensation, un critère supplémentaire, le but du dispositif, est aussi présent.

orthèses cognitives. Néanmoins, la comparaison et surtout la compréhension de ces outils ne peut se limiter à l'étude de tels critères, purement statiques. En effet, le processus d'assistance cognitive est avant tout un processus dynamique : c'est au cours de la réalisation de ses AVQ que les besoins de la personne vont s'exprimer. C'est donc aussi sous cette dimension que doivent être étudiées les orthèses cognitives, afin d'exprimer de façon plus complète le rapport entre la technologie et l'assistance cognitive. En exprimant cette relation, on souhaite pouvoir expliquer le fonctionnement de l'outil d'assistance cognitive étant donné un contexte particulier d'assistance cognitive. Ainsi, l'adéquation entre l'outil et les besoins d'assistance cognitive pourra être discutée, voire pensée si aucun outil ne répond aux besoins.

Dès lors, est proposé dans la première section de ce chapitre un *cadre fonctionnel pour l'étude et la conception des orthèses cognitives*, formalisant le principe d'une assistance cognitive basée sur la technologie. Ce cadre fonctionnel est par la suite appliqué à l'étude de différentes orthèses présentes dans la littérature. Dans un premier temps, la présentation couvre les outils mobiles les plus significatifs. Cette discussion met en avant les limitations de ces outils lorsque l'assistance doit être plus poussée. Cela ouvre la porte à la présentation d'une autre approche technologique, basée sur le paradigme de l'informatique ubiquitaire. Cette notion et son application à l'assistance cognitive font l'objet de la quatrième section du chapitre. Celle-ci est suivie par quelques réflexions conclusives sur l'utilisation de la technologie à des fins d'assistance cognitive. Le tout ouvre la voie à la présentation, au chapitre 3, des fondements de l'orthèse Archipel, objet du présent travail.

2.1 Un cadre fonctionnel pour l'étude et la conception des orthèses cognitives

Imaginons une montre avec alarme, utilisée pour rappeler la prise de médicaments, et un habitat intelligent, capable de rendre à une personne fonctionnellement limitée sa parfaite autonomie. Peut-on dire qu'il existe une analogie entre ces dispositifs ? Oui et non. Oui, car tous deux prennent place dans un même contexte applicatif : celui de l'assistance cognitive. Non, car la connaissance, la perception, la compréhension et l'utilisation qu'ils ont et font de ce contexte est, pour des raisons techniques, fondamentalement différente. Ils n'abordent pas l'assistance cognitive de la même façon, et ne seront donc certainement

pas capables de répondre aux mêmes besoins.

Pour la montre, le rappel de la prise correspond à une sonnerie réalisée à heure précise et de façon systématique. On résume ainsi :

1. sa connaissance des besoins de la personne, soit le fait de devoir effectuer un rappel à une heure donnée ;
2. sa perception du contexte dans lequel prend part l'assistance cognitive, qui se concrétise par sa capacité à mesurer l'écoulement du temps ;
3. sa compréhension de l'écoulement du temps, que l'on peut résumer par *si heure courante égale heure du rappel, alors "faire sonner" sinon rien* ;
4. l'utilisation faite de ses ressources technologiques, soit lorsque cela est approprié le déclenchement de la sonnerie.

Si le même rappel est confié à l'habitat intelligent, on peut imaginer que celui-ci soit localisé en fonction de l'emplacement courant de la personne, et réalisé selon une modalité adaptée aux préférences de cette dernière. Par rapport à la montre, l'habitat doit disposer en complément (1) de la connaissance des préférences et des dispositifs de rappel présents dans l'habitat, (2) d'une capacité de perception et (3) d'interprétation des déplacements de la personne, et (4) d'une capacité de sélection d'un dispositif de rappel adapté au contexte (préférence, emplacement). On peut aussi aller beaucoup plus loin, et supposer que l'habitat peut percevoir et interpréter la prise des médicaments, lui permettant d'aviser la personne si le moment de la prise n'est pas adéquat, si celle-ci a été déjà réalisée, *etc.*

Dès lors, parler de *cadre fonctionnel pour l'étude et la conception des orthèses cognitives* revient à définir deux aspects. Le premier correspond à l'ensemble des concepts qui décrivent le *contexte d'assistance cognitive (CAC)*, et dont font notamment partie les notions de *besoins*, de *préférences*, d'*événements* d'écoulement du temps ou de déplacement. Le second concerne la capacité de l'orthèse à disposer de ces données contextuelles. On parlera alors de *sensibilité au contexte d'assistance cognitive*, qui regroupe la *perception*, la *compréhension* et l'*utilisation* du CAC. Enfin, si le CAC est une composante universelle, la sensibilité au CAC est propre à chaque orthèse, et dépend notamment de sa capacité à *représenter* les connaissances contextuelles.

Vers la production d'*actes d'assistance* L'objectif de cette mécanique fonctionnelle est claire : assister la personne sur un plan cognitif. Dans un contexte purement

humain, où la personne est assistée d'un aidant, le rappel de la prise de médicaments se manifeste certainement par un énoncé oral. Il s'agit alors d'un *acte de langage*, dans la mesure où ces paroles ne se limitent pas à énoncer un fait mais sous-entendent (suggèrent ou commandent) la réalisation d'une action [5]. En s'inspirant de cette notion, on définira par *acte d'assistance cognitive* l'ensemble des mises en oeuvre réalisées par l'orthèse, quelle qu'en soit la nature, destinées à répondre à un besoin d'assistance du client, correspondant à la compensation d'un trouble cognitif. Il pourra tout aussi bien s'agir de la production d'un message vocal que d'une intervention directe sur un artefact de l'environnement (couper l'alimentation en eau d'une baignoire qui risque de déborder) ou d'un message envoyé à un tiers. Les actes d'assistance représentent la pointe émergée de l'iceberg de l'assistance cognitive, la mécanique sous-jacente restant transparente pour l'utilisateur.

D'après le dictionnaire Nouveau Petit Robert 2007, le "contexte" désigne l'*ensemble des circonstances dans lesquelles s'insère un fait*, ces circonstances étant généralement significatives pour comprendre la sémantique du fait. Les faits qui intéressent ce travail sont les actes d'assistance cognitive, dont la justification de la (non-)survenance et de la sémantique se trouve dans le CAC et la sensibilité de l'orthèse au CAC. Ainsi, le cadre fonctionnel doit être vu comme un outil de réflexion sur le fonctionnement de l'orthèse. Mais c'est dans la conception qu'il devrait être le plus utile, permettant de trouver la meilleure adéquation entre besoins d'assistance et ressources technologiques.

2.1.1 Le Contexte d'Assistance Cognitive (CAC)

La première facette de ce cadre fonctionnel concerne donc l'ensemble des circonstances dans lesquelles s'insèrent les actes d'assistances, soit le *Contexte d'Assistance Cognitive (CAC)*. Le CAC prend lui même place dans un contexte plus large, celui des faits de toute nature. La définition de ce sous-ensemble contextuel est avant-tout pédagogique : il s'agit d'identifier et de nommer les différents concepts contextuels liés à l'assistance cognitive, afin de faciliter la description fonctionnelle de l'orthèse.

L'exemple de la montre-alarme utilisée pour rappeler la prise de médicaments introduit deux éléments du CAC. Le premier est le besoin de rappel, exprimé par la programmation à une heure donnée de l'alarme, le second l'écoulement du temps. Tous deux sont nécessaires pour que la montre produise l'acte d'assistance de rappel.

Ces deux éléments se différencient par leur persistance dans le temps. Le besoin de

rappel est relativement statique : on peut en effet supposer que la prise va intervenir à la même heure chaque jour. Une évolution restant possible, on qualifiera cet aspect du CAC de *semi-statique.* L'écoulement du temps est pour sa part clairement dynamique : à chaque seconde, si l'on suppose que cette unité est appropriée au fonctionnement humain, le contexte lié à l'écoulement du temps change. Dès lors, on gardera l'appellation de *dynamique* pour qualifier cet aspect du CAC.

A défaut de mieux, on se dirige vers une définition du CAC par l'exemple. Néanmoins, pour en assurer la complétude, on se basera sur la revue la plus récente du domaine des orthèses cognitives [121], dont on s'efforcera d'extraire et de formaliser les caractéristiques fonctionnelles. On précisera que cette revue est envisagée sous l'angle des besoins des utilisateurs, suivant la classification proposée par D. Maciuszek [98]. Ce dernier définit un ensemble de *pattern of assistive interaction, i.e.* d'approches conceptuelles qu'une orthèse peut mettre en oeuvre pour répondre à un ou plusieurs besoins. Ainsi, le patron *reminder* - rappel - est mis en oeuvre par la montre pour répondre au besoin découlant de troubles mnésiques. Le patron *guide* doit pour sa part se concrétiser par une assistance pas à pas pour la réalisation d'activités, faisant ainsi écho à des problèmes mnésiques procéduraux ou de planification. Au total, quinze patrons de ce type ont été isolés par Maciuzsek.

CAC *semi-statique*

CAC *humain.* Par CAC *humain,* on désigne l'ensemble des individus impliqués dans le processus d'assistance et leurs caractéristiques du point de vue de l'assistance. Au premier plan se situe le ou les client(s), directement concernés par les actes d'assistance cognitives. Ces personnes ont un *profil cognitif,* duquel découle un ensemble de *besoins.* Pour répondre à ces besoins, un *profil d'assistance* doit être établi. Ce second profil concerne l'approche à mettre en oeuvre pour répondre à ces besoins, qui dépend du patron choisi (dans l'esprit de Maciuzsek), des préférences de la personne et autres considérations objectives la concernant, comme les données physiques (capacités visuelles et auditives, *etc.*) ou psychologiques (niveau de fatigue, *etc.*). Au second plan du contexte humain gravitent l'ensemble des tiers qui sont en relation avec le client. Il peut s'agir d'aidants naturels ou professionnels, dont le *rôle* doit être précisé pour pouvoir rentrer dans la boucle d'assistance [32].

A noter que la notion de *besoins* introduite ici correspond aux besoins *exprimés* par la personne, ou l'un de ses tiers. On adopte ici le point de vue de la personne, et non celui de

l'orthèse. Néanmoins, il doit exister une relation forte entre les besoins exprimés et ceux qui vont être ciblés par l'orthèse, Baecker parlant de *processus cognitifs ciblés*. La force de cette relation correspond au niveau d'adéquation existant entre l'un et l'autre, un des enjeux du développement d'orthèse. Dans le cadre fonctionnel présenté ici, le point de vue de l'orthèse, celui des besoins *ciblés*, est une composante transversale de la sensibilité au CAC, influençant clairement la mise en oeuvre de ses différentes facettes.

CAC *temporel.* Si les besoins exprimés déterminent qu'un rappel des activités doit être réalisé pour le client, c'est le contexte temporel qui précisera les contraintes temporelles inhérentes aux activités. D'une manière générale, la notion du temps est très présente dans les orthèses cognitives. Les télé-avertisseurs, qui figurent parmi les premiers outils technologiques à avoir été largement utilisés en clinique, se bornent à effectuer à heure prévue le rappel d'activité [67, 147]. Les logiciels grand public des PDA, présents depuis plusieurs années en réadaptation, supportent l'organisation du quotidien (emploi du temps avec rappel des rendez-vous, *etc.*) [84]. Cette question de l'aide à la gestion du quotidien a aussi trouvé écho dans divers logiciels développés spécifiquement pour la clientèle. C'est le cas avec PEAT [91], MEMOS [136] ou Autominder [127], dont les techniques de planification temporelle mises en oeuvre pour la compréhension/utilisation du contexte seront présentées ultérieurement. Enfin, OrientingTool [160] s'efforce de replacer les personnes présentant une amnésie antérograde[2] dans un contexte temporel et spatial, en leur permettant de noter la raison pour laquelle ils se trouvent à un endroit à un moment donné.

CAC *matériel.* Par CAC *matériel*, on entend les artefacts présents dans l'environnement de la personne mais ne faisant pas partie de l'orthèse. On pourrait les qualifier d'objets de la vie quotidienne, jouant souvent un rôle de premier plan lors de la réalisation des AVQ. Ainsi dans COACH, une orthèse dédiée à l'assistance au lavage des mains, la serviette et le savon sont intégrés au processus de suivi de la réalisation de l'activité [103]. C'est aussi le cas pour les activités prenant place en cuisine, où les objets sont nombreux, spécifiques et indispensables à la réalisation [96].

CAC *technologique.* Le CAC *technologique* comprend l'orthèse, au sens large. Ainsi, on regroupera sous cette appellation non seulement l'"unité pensante", par exemple la

[2]Incapacité à acquérir des faits nouveaux. Ainsi la présence dans un lieu inconnu peut entraîner un état de panique, la personne étant incapable de se repérer et de se rappeler ce qu'elle est venue faire ici.

ou les machines sur lesquelles fonctionnent les programmes informatiques dédiés à l'assistance cognitive, mais aussi tous les composants plus ou moins inertes distribués dans l'environnement et représentant les possibles extensions de ces unités. Comme on le verra par la suite, les contacts électromagnétiques, étiquettes et lecteurs à radio-fréquence, débit-mètre, écran tactile et autres dispositif lumineux contrôlés par courant porteur sont quelques exemples de *capteurs* et d'*effecteurs* pouvant jouer un rôle dans le processus d'assistance cognitive [16, 120]. Dès lors, un des enjeux du développement consiste en la *représentation* de ces données contextuelles, afin d'offrir au programme coeur de l'orthèse la possibilité de manipuler ses différentes composantes.

D'une manière générale, il existe un lien étroit entre le CAC technologique et la sensibilité au contexte. Si l'on se place du point de vue de l'étude et de la compréhension des orthèses cognitives, l'explication de la capacité d'une orthèse à percevoir le contexte se trouvera dans ses caractéristiques technologiques. Par exemple, la présence de détecteurs de mouvements permettra à l'outil de percevoir les déplacements de la personne. Le raisonnement s'effectue donc du CAC technologique vers le processus de sensibilité, en l'occurrence les capacités de perception. On rejoint la notion de type de technologie citée par Baecker. Si par contre on travaille sous l'angle du développement, de la nécessité de percevoir certains éléments contextuels découlera celle de disposer de ressources technologiques particulières. C'est toute l'idée de ce cadre fonctionnel.

Pour terminer, on notera que certains objets peuvent se retrouver à cheval entre le matériel et le technologique, s'ils ont la capacité de servir de média d'interaction entre la personne et l'orthèse. On pense par exemple aux interfaces saisissables, soit un objet physique dont la manipulation matérielle se confond avec celle d'une fonctionnalité virtuelle du système [53]. Cet aspect sera discuté au chapitre 5.

CAC *spatial*. Le CAC *spatial* concerne la configuration des lieux où prennent place les activités, ainsi que la localisation et l'identification des humains et des artefacts matériels [129]. Grâce au système de positionnement par satellites GPS [3], la localisation en extérieur des individus a été largement intégrée dans les orthèses mobiles dédiées à l'aide aux déplacements comme Opportunity Knocks [118], MAPS [30] et MOBUS [124]. Du côté identification, la technologie d'identification par radio-fréquence (RFID, *Radio Frequence IDentification*) connaît un essor particulier avec l'incorporation à une échelle de plus en plus grande d'étiquettes d'identification aux objets du quotidien, et ce dès leur

[3] *Global Positioning System.*

fabrication. On peut ainsi envisager de les suivre individuellement tout au long de leur cycle de vie [128].

CAC *dynamique*

Les aspects dynamiques du CAC, comme l'écoulement du temps, morcellent le contexte en un ensemble de *situations contextuelles*. Chaque situation témoigne de la survenance d'un nouvel événement lié à la dynamique du CAC. Par exemple, s'agissant du temps, chaque passage de seconde, si telle est l'unité retenue, sera à l'origine d'une nouvelle situation contextuelle.

Trois types d'événements doivent être considérés :

Les événements naturels. Il s'agit des événements sur lesquels ni l'homme ni la technologie n'a d'autorité. L'écoulement du temps en est le principal exemple ;

Les événements humains. Ces événements découlent du comportement des personnes et sont supposés volontaires. Rentrent dans cette catégorie les événements de déplacement, ainsi que les interactions avec les autres personnes, la technologie et le matériel ;

Les événements technologiques. Cette catégorie comprend les événements dont la technologie d'assistance est explicitement à l'origine, et qui correspondent aux actes d'assistance.

2.1.2 Sensibilité au Contexte d'Assistance Cognitive

Le deuxième aspect de ce cadre fonctionnel est la *sensibilité au CAC*, plus précisément les mécanismes menant à la prise en compte à des fins d'assistance cognitive du CAC semi-statique et dynamique. On distingue ici trois facettes, la *perception* du CAC, sa *compréhension* et son *utilisation*, formant une boucle de contrôle naturelle mais non contraignante.

Par *perception*, on entend la capacité d'une orthèse à recevoir les événements du CAC dynamique. La *compréhension* correspond à l'étape subséquente visant à composer avec l'ensemble des événements reçus et les situations antérieures pour fixer la sémantique de la situation courante. Une *utilisation* peut alors être faite du CAC pour produire un acte d'assistance cognitive.

Bien entendu, le degré de perception, de compréhension et d'utilisation dépend du CAC, plus précisément des capacités globales de l'orthèse. Ainsi il faut bien détacher le CAC de la conscience qu'en a une orthèse : de nouveaux événements peuvent se produire

et de nouvelles situations se mettre en place sans que l'orthèse y soit sensible, tout simplement parce que les outils dont elle dispose pour percevoir le CAC ne le permettent pas. Mais peu importe, l'essentiel est que l'orthèse puisse fournir l'assistance appropriée à son ou ses clients, autrement dit qu'elle soit *utilisable*. L'utilisabilité des dispositifs existants sera discutée dans la suite de ce chapitre, mais auparavant, les exemples d'événements et d'actes d'assistance présentés à la figure 2.1 permettront d'illustrer les diverses notions de ce cadre fonctionnel. En voici l'explication :

Evénement *x* Au temps x survient un événement d'écoulement du temps que l'orthèse perçoit. La compréhension qu'en a l'orthèse est guidée par les besoins du client, plus particulièrement en terme de rappel d'activités : il est en effet l'heure de notifier la prise de médicaments. L'orthèse utilise alors le contexte technologique (capacités d'interaction) et humain (profil d'assistance) pour effectuer judicieusement ce rappel. C'est l'acte d'assistance ★1 ;

Evénement *x+2* L'orthèse ne perçoit pas l'événement $x+2$, elle n'a donc pas connaissance de la situation $x+2$;

Evénement *x+3* A $x+3$, la personne demande explicitement de l'assistance à l'orthèse. Cette demande est traitée conformément au besoin en cours (rappel de la prise de médicaments) et au profil d'assistance de la personne. Par exemple, l'orthèse pourrait proposer une aide pas à pas pour la réalisation de l'activité. C'est ce qu'elle fait avec l'acte d'assistance ★2, en utilisant le contexte technologique pour diffuser cette information ;

Evénement *x+5* Alors qu'il réalise l'activité de prise de médicaments, l'orthèse perçoit que le client utilise un objet non adéquat voire dangereux pour lui, ainsi que le précise le contexte matériel et humain. Il lui en fait part (★3), en tenant compte de l'acte d'assistance précédent, puis met à jour le profil d'assistance afin d'éviter qu'une telle erreur se répète.

Evénement *x+8* Un intervenant interagit avec le système pour mettre à jour le profil cognitif du client, qui s'est manifestement dégradé. L'orthèse s'assure toutefois que la qualité de ce tiers lui permette cette intervention.

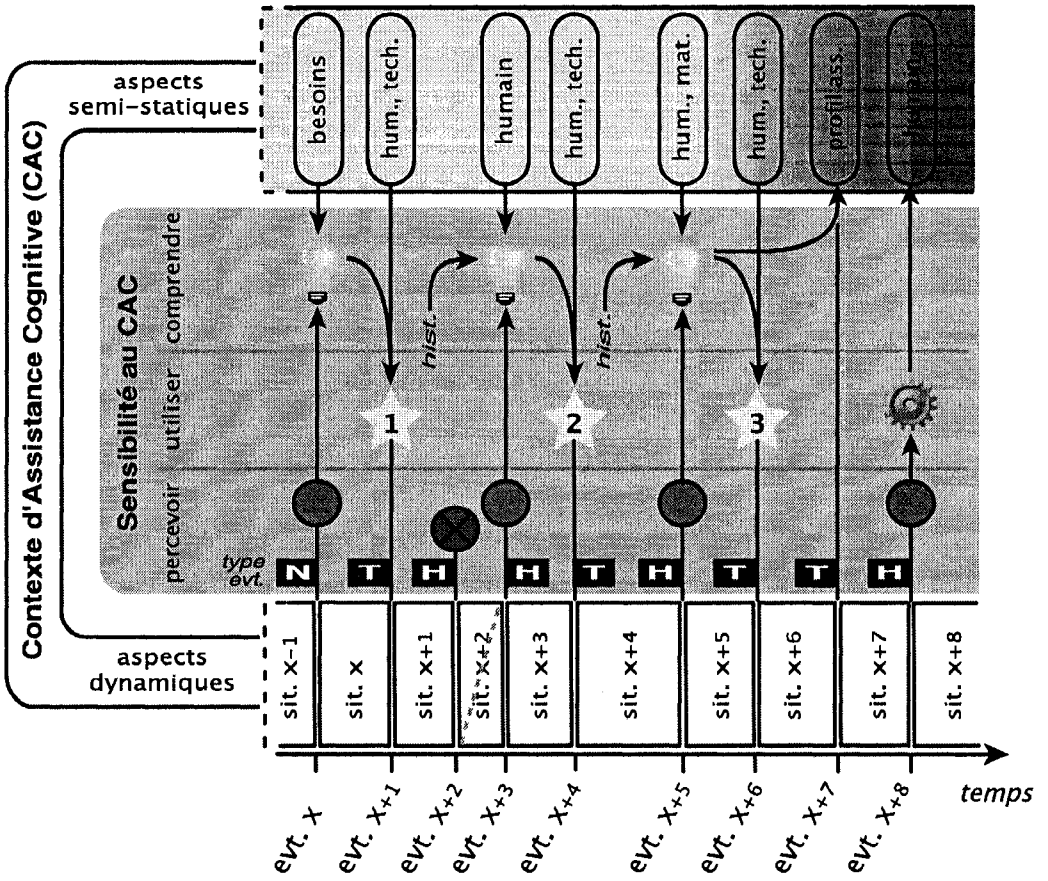

FIGURE 2.1 – Cadre fonctionnel pour les orthèses cognitives : illustration de la sensibilité au Contexte d'Assistance Cognitive. *hist.* représente l'historique des situations contextuelles. Les lettres sur fond noir N, T, et H précisent le type de l'événement : naturel, technologique ou humain.

2.2 Du cadre fonctionnel à la mise en oeuvre

2.2.1 Quelques généralités

Décrire le CAC. Il a déjà été dit que la représentation des données contextuelles était une étape incontournable de la mise en oeuvre d'une orthèse, afin que celle-ci puisse percevoir, comprendre et utiliser le CAC. Cette représentation peut être matérielle, comme dans les composants d'une montre à quartz, ou logicielle, ouvrant ainsi la porte à une représentation d'une grande richesse et à une sensibilité d'une toute autre nature. La discipline alors invoquée est la *représentation des connaissances (RC).*

Dans leur article *What is Knowledge Representation ?* [43], R. Davis *et al.* définissent cinq rôles joués par cette discipline qui sont autant d'objectifs à atteindre et donc de problématiques à résoudre pour qu'une approche de la RC soit effective :

- une RC est avant-tout un substitut du monde réel permettant de raisonner sur celui-ci sans passer par les actes ;
- c'est une vision partiale du monde, justifiée par les objectifs de la représentation ;
- c'est un espace fini de raisonnement, basé sur (1) ce qu'est le raisonnement du point de vue de cette représentation, (2) les inférences qu'elle permet de faire et (3) les inférences qu'elle préconise ;
- c'est un support devant permettre la réalisation efficace, du point de vue informatique, des inférences préconisées ;
- c'est enfin un langage permettant aux humains de discourir du monde.

Il existe divers courants informatiques de représentation des connaissances. Le chapitre 4 en fera écho et détaillera les choix pris pour Archipel, dont la justification repose, on l'aura compris, dans la sensibilité que l'on veut donner à cette orthèse pour répondre à un ensemble défini de besoins.

Percevoir le CAC. Concernant la perception du contexte, deux aspects doivent être envisagés. Le premier est la *perception interne*, c'est à dire celle des événements dont l'orthèse est elle-même à l'origine. L'exemple le plus trivial est celui de l'écoulement du temps. Si une application logicielle met en oeuvre sa propre horloge, il semble évident qu'elle percevra les événements d'incrémentation de celle-ci. Néanmoins, cette facette de la perception reste, du point de vue fonctionnel, des plus importantes. L'autre aspect est la *perception externe*, celle des événements dont l'orthèse n'est pas à l'origine. Le facteur humain rentre alors en ligne de compte, source d'événements correspondant aux actions

réalisées par la personne dans son environnement. La problématique ciblée ici est celle de l'*interaction homme-machine (IHM)*, sous ses facettes explicites et implicites, selon que la finalité première de l'acte soit de communiquer ou non avec le système.

Les interactions implicites sont, comme on le verra par la suite, souvent perçues grâce à des supports technologiques de type capteur. Se pose alors la question du niveau d'abstraction de l'interaction, au titre de la perception du CAC. En effet, pour les dispositifs les plus simples, les données reçues sont brutes, du type "ouverture/fermeture d'un circuit électrique" pour un contact électromagnétique installé sur une porte. Est-ce là une information d'IHM, ou est-ce l'interprétation de cette ouverture/fermeture de circuit sous la forme d'une action de la personne (ouverture/fermeture d'une porte donnée) qui constitue l'interaction et donc la perception du CAC ? La différence réside en un certain nombre de traitements intermédiaires visant à contextualiser l'information. On posera ici que la perception du CAC est celle d'événements contextualisés mais atomiques, i.e. dont la granularité est minimale dans le contexte de perception donné. On pose ainsi la limite entre la perception et la compréhension du CAC, qui aura notamment pour charge l'intégration à des fins d'interprétation des événements atomiques. Le chapitre 5 reviendra sur ces notions d'IHM et sur la mise en oeuvre qui en est faite dans Archipel.

Comprendre le CAC. Comprendre, c'est *être capable de faire correspondre à quelque chose une idée claire*[4]. Pour l'assistance cognitive, le "quelque chose" est un ensemble d'événements perçus par l'orthèse, donnant naissance à tout autant de situations contextuelles. Etant donné un CAC, le problème est alors de faire correspondre à cette séquence de situations une logique d'assistance.

Dans l'exemple de la montre, la compréhension se modélise par un simple test conditionnel : si la nouvelle situation correspond d'un point de vue temporel à l'heure souhaitée pour le rappel, alors effectuer ce rappel. A l'autre extrémité, la compréhension peut se traduire par le suivi de la réalisation d'une activité complexe, l'orthèse devant reconstruire le cheminement de réalisation en fonction des situations contextuelles et du CAC, afin de s'assurer de son effectivité et de détecter et qualifier les éventuelles erreurs, pour pouvoir y répondre. Dans la littérature, ce type de problématique est généralement associé à une question de reconnaissance d'activité (ou de plan), discipline de l'intelligence artificielle établie et qui fait l'objet d'une recherche active [25]. Dans ce travail, ce problème de compréhension sera abordé sous l'angle du monitorage d'activité, tel qu'introduit au

[4]Définition du dictionnaire Nouveau Petit Robert 2007, 2ème acception

chapitre 6. Quoi qu'il en soit, l'essence même de la compréhension est d'offrir le support nécessaire pour répondre à un besoin d'assistance exprimé, tel qu'effectuer un rappel ou assurer le suivi d'une réalisation.

Au final, la compréhension s'inscrit comme l'étape charnière du processus de sensibilité au CAC. Elle nécessite la perception des événements, et peut engendrer l'utilisation du CAC à des fins de production d'un acte d'assistance (rappel, réponse à une erreur de réalisation, *etc.)*. Le tout forme une boucle naturelle mais non contraignante de contrôle pour ce processus.

Utiliser le CAC. L'acte d'assistance est la partie visible de l'iceberg de l'assistance cognitive ; produire cet acte, c'est avant tout utiliser le CAC. Ainsi, si la production fait clairement référence à une problématique d'interaction machine-homme, sa génération couvre de nombreux aspects du CAC, afin de personnaliser l'acte et sa production pour en assurer une meilleure compréhension. Les préférences de l'utilisateur, son profil d'assistance, sa compréhension des actes précédents, ses caractéristiques spatiales et les possibilités technologiques sont quelques aspects qu'une orthèse pourra tenter de combiner afin de produire subtilement l'acte, si telles sont ses capacités. A noter que cette réflexion peut parfaitement aller jusqu'à la non-production de l'acte malgré la détection d'une erreur, notamment si la sémantique de cette dernière le justifie. Dans les cas les plus simples, la production se concrétisera par l'utilisation sans réflexion supplémentaire de l'unique canal de communication, comme c'est le cas pour la montre-alarme.

Mais l'utilisation peut aussi être un processus pro-actif, avec l'ajout ou l'ajustement, par un mécanisme d'apprentissage, de la connaissance qu'a l'orthèse de son client. Pour l'orthèse Opportunity Knocks, l'apprentissage concerne la connaissance des déplacements usuels de la personne, afin de mieux la guider par la suite [118]. L'outil peut aussi tenter d'anticiper le comportement de la personne, avec la mise en oeuvre de techniques issues du domaine de l'informatique cognitive [137]. De tels modèles comportementaux permettraient d'envisager, à long terme, une évaluation *a priori* de l'impact de tel ou tel acte d'assistance sur l'utilisateur. Toutefois, dans ce travail, on se limitera aux aspects génération de l'acte d'assistance, comme présenté au chapitre 7.

2.2.2 L'utilisabilité comme critère d'évaluation

Evaluer une orthèse cognitive, c'est l'expérimenter auprès de la clientèle. Au chapitre 8 sera présentée l'expérimentation réalisée pour Archipel, auprès de douze adultes

présentant une déficience intellectuelle. Mais avant cela, il est intéressant de se poser quelques questions sur les critères permettant d'évaluer l'atteinte des objectifs d'assistance d'une orthèse.

Globalement, une orthèse doit (1) répondre aux besoins de la personne, et (2) y répondre de façon adéquate. En effet, s'il est absolument nécessaire, le premier de ces critères n'est pas suffisant. L'exemple de la montre-alarme l'illustre parfaitement : si le besoin du client s'exprime en terme de rappel d'activité, alors un tel dispositif offre une réponse. Néanmoins, sa mise en oeuvre va généralement se matérialiser par une sonnerie, ce qui signifie que l'efficacité de l'acte d'assistance repose sur la capacité de la personne à faire le lien entre cette sonnerie et l'activité à réaliser. Or certains profils cognitifs ne permettent pas une telle association sémantique [125], confirmant que la façon d'apporter la réponse est partie intégrante du processus d'assistance. Bien qu'évidente, cette remarque pose un véritable défi pour la clientèle présentant des troubles cognitifs.

Il est possible de rapprocher ces deux critères d'une notion plus générale, celle de l'utilisabilité. Du point de vue normatif, un système est dit utilisable s'il "permet à son utilisateur de réaliser sa tâche avec efficacité, efficience et satisfaction dans le contexte d'utilisation spécifié" [73]. Par efficacité, on entend que l'utilisateur puisse atteindre ses objectifs, grâce à une utilisation mesurée (efficience[5]) et agréable (satisfaction) du système et des ressources. Cette notion est notamment appliquée en informatique pour la conception d'interfaces utilisateurs, la facilité d'apprentissage de l'application rentrant aussi en ligne de compte [107].

Une orthèse sera utilisable s'il existe un lien étroit entre les besoins et spécificités du client et les capacités de l'orthèse, basées sur ses caractéristiques technologiques et sa sensibilité au CAC. L'utilisabilité des orthèses revêt donc un caractère profondément subjectif, et il est difficile de préjuger, dans l'absolu, de l'intérêt d'une orthèse. Ainsi la simple montre-alarme pourra parfaitement s'avérer utilisable pour une certaine clientèle. Néanmoins, les orthèses plus évoluées vont tendre, comme on le verra, vers une clientèle plus universelle, grâce à diverses possibilités de personnalisation. On se rapproche des critères de E. Cole, posés en introduction de ce chapitre [35, 36].

Afin de donner une première idée des solutions proposées par les chercheurs et cliniciens pour répondre à cet objectif d'utilisabilité, un portrait global du domaine est dressé dans les deux prochaines sections. Dans un premier temps, on s'intéresse aux

[5]Le temps est souvent cité comme exemple de ressource pour la mesure de l'efficience.

orthèses mobiles, qui représentent les dispositifs technologiques d'assistance cognitive les plus démocratisés. Les limites de ces orthèses en terme de connaissance implicite des difficultés de l'utilisateur invitent dans un second temps à s'interroger sur l'application à l'assistance cognitive d'une informatique plus ubiquitaire.

2.3 Analyse des orthèses existantes : les solutions mobiles

Les assistants numériques personnels ou PDA[6] ont une place prépondérante dans le domaine des orthèses cognitives. Mobile et portable, disposant d'une mémoire et d'une unité de traitement dont les capacités sont aujourd'hui bien supérieures aux premiers ordinateurs personnels, le PDA semble à même d'offrir un support considérable pour la compréhension et l'utilisation du CAC. En fait, le PDA et ses applications logicielles s'inscrivent comme l'équivalent technologique de l'agenda "papier-crayon", outil traditionnellement employé en réadaptation pour répondre à certaines problématiques mnésiques et exécutives [121]. Sa mobilité et sa taille en font un dispositif consultable en tout temps et tout lieu. Dans le cas des *smartphones*, les capacités étendues de communication, *via* l'accès aux réseaux de téléphonie mobile, ajoutent à l'outil la possibilité de se connecter à divers services distants, et de communiquer aisément avec les tiers. Le tout renforce clairement l'intérêt que l'on peut avoir pour ce type d'outil. L'interaction homme-machine reste par contre limitée à l'écran tactile, de petite taille, et éventuellement à un similiclavier. Ainsi, la perception des besoins ne peut être qu'explicite, et c'est certainement là sa principale limitation.

C'est aussi un objet largement répandu dans la population. Sa très grande acceptation sociale en fait un outil normalisant voire valorisant. Ces considérations ont leur importance en clinique : la personne doit accepter son orthèse, et au même titre qu'un fauteuil roulant colle à son utilisateur une étiquette "handicapé", avoir comme support mnésique un énorme classeur n'a rien d'engageant. Ce n'est pas la vision qu'offre un PDA [121], et l'utilisabilité ne peut en être que valorisée.

[6] *Personal Digital Assistant.* L'acronyme anglais est d'usage.

2.3.1 Applications grand public et assistance cognitive

Du côté des besoins, selon la classification de Maciuzsek [98], les applications grand public des PDA s'inscrivent comme support pour la mémoire prospective, avec les gestionnaires de rendez-vous et de tâches, assurant stockage de l'information et diffusion de signaux externes incitatifs. Ils s'inscrivent d'autre part comme support mnésique à long terme, avec les notes et les contacts pour la rétention de concepts sémantiques, et les rendez-vous passés comme éléments de souvenirs. Cependant, le grand nombre de possibilités offertes par ces outils grand public augmente leur complexité d'utilisation : ces possibilités dépassent les besoins de la clientèle et entravent l'apprentissage [35,83,136]. Cette complexité se retrouve au niveau des interfaces graphiques, qui, en amont, nécessitent de trop nombreuses étapes pour saisir l'information, et, en aval, n'offrent pas d'accès direct à l'information stockée [60, 147]. Dès lors, c'est à l'utilisateur d'aller vérifier l'existence d'informations précédemment saisies, ce qui est inadapté à la clientèle avec troubles cognitifs [143]. De même, les capacités de personnalisation, comme l'ajustement de la taille des boutons ou des caractères, sont restreintes [121]. Néanmoins, malgré ces limites manifestes à l'utilisabilité, les études précitées révèlent une augmentation de l'indépendance des utilisateurs, l'outil permettant une meilleure gestion de leur quotidien.

2.3.2 Les approches dédiées à la clientèle

Historiquement, le premier besoin auquel les chercheurs se sont intéressés est le rappel d'activités. Le support technologique alors utilisé est un télé-avertisseur côté utilisateur couplé à une base de donnée centralisée. Celle-ci contient l'information liée aux rappels et peut être indirectement mise à jour par les aidants[7] [67]. Nommé Neuropage, ce service de rappel a été largement utilisé et testé au Royaume-Uni [158]. Les résultats montrent une augmentation du nombre d'activités réalisées par rapport au nombre d'activités ciblées, passant de moins de 50% à environ 76%. Une fois l'appareil retiré, on note une diminution du nombre d'activités réalisées (62%), restant malgré tout supérieur au taux initial[8]. Aujourd'hui des sociétés commerciales fournissent des services similaires[9].

Le télé-avertisseur reste néanmoins un simple afficheur alpha-numérique avec des capacités de communications, tout du moins de réception de l'information. Son utilisation

[7]Chaque demande de rappel doit être transmise à l'avance à un opérateur pour traitement..

[8]Résultat confirmé dans d'autres études, qui tend à prouver non seulement le caractère compensatoire des orthèses cognitives, mais aussi leur apport en terme de restauration des fonctions cognitives déficientes

[9]www.memotome.com ; www.pageminderinc.com

en tant qu'orthèse cognitive est donc limitée à un type très ciblé de besoin. Les moyens de traitement des PDA ont permis le développement d'outils plus évolués, que l'on qualifiera de gestionnaires d'emploi du temps. Les actes d'assistance vont alors du rappel à l'aide à la réorganisation de l'horaire : ajout, report d'une activité, ou encore prolongement de la durée de l'activité en cours. Les problématiques en jeu sont la gestion de contraintes temporelles et de la priorité entre activités, thématiques complexes pour la clientèle. Les applications pour PDA, PEAT (Planning and Execution Assistant and Trainer) [90][10] et MEMOS (Mobile Extensible Memory Aid System) [136], utilisent des techniques d'intelligence artificielle et plus particulièrement de planification pour aborder ces problèmes et conseiller l'utilisateur. L'utilisation du CAC prend une toute autre tournure, soutenue en cela par une représentation nécessairement plus riche des activités à rappeler (voir chapitre 7). A noter l'interface graphique très simple de MEMOS (liste d'activité, fenêtre de validation pour la réorganisation de l'emploi du temps), et les options de personnalisation de PEAT. Ce dernier proposant aussi un gestionnaire de note et de contact, il est ainsi possible de désactiver certaines fonctionnalités, l'interface se simplifiant d'autant.

On nommera la deuxième problématique abordée par les solutions dédiées "guide de réalisation d'activités". L'idée communément admise ici est de présenter dans un mode pas à pas la séquence des étapes de réalisation - procédure - pour un objectif présélectionné, selon différentes modalités. Avec ISAAC[11], la présentation est textuelle et orale (synthèse vocale) [60]. Pour le Visual Assistant d'Ablelink[12], la présentation se fait selon une modalité multimédia, utilisant le son et l'image. L'utilisabilité d'ISAAC est assez exemplaire, grâce à un accès efficace à l'information, tant sur le plan physique que cognitif. Les procédures de réalisation sont accessibles par le biais de boutons dont la taille est ajustable, afin que la sélection puisse être réalisée avec le doigt plutôt que le stylet. Chaque bouton représente une catégorie explicite d'informations, l'organisation des catégories étant hiérarchique (catégorie, sous-catégorie, procédure). L'évaluation réalisée auprès de deux utilisateurs a donné des résultats significatifs, avec un impact bénéfique sur leur indépendance fonctionnelle [60]. Les évaluations cliniques du Visual Assistant, menées auprès de personnes présentant une déficience intellectuelle, montrent une diminution significative de la demande d'aide extérieure et du nombre d'erreurs lors de la réalisation d'activités simples [42].

[10]www.brainaid.com - commercialisé, langue anglaise.

[11]www.cosys.us - commercialisé.

[12]www.ablelinktech.com. AbleLink propose d'autres outils pour la clientèle présentant une déficience intellectuelle.

Le dernier aspect significatif pour les orthèses portables est l'aide à la mobilité. Deux problématiques sont ici abordées : aider aux déplacements et garder le lien avec l'aidant. Opportunity Knocks, de l'Université de Washington à Seattle [118] et MAPS (Memory Aid Prompting System) [29, 30] proposent l'aide aux déplacements. Tous deux utilisent un récepteur GPS[13] pour connaître l'emplacement de l'utilisateur. Cette information est alors utilisée pour inférer la bonne position de la personne par rapport à une sélection effectuée parmi un ensemble de chemins préétablis, et réagir en conséquence. On franchit ainsi le cap d'une perception implicite du contexte d'assistance cognitive, en l'occurrence sous sa facette spatiale : l'acquisition des données concernant les déplacements de la personne est réalisée de façon automatique, sans intervention de l'utilisateur.

Cet enrichissement de la connaissance du CAC permet à Opportunity Knocks d'aller plus loin quant à son utilisation, avec un apprentissage des trajets usuels de la personne. Couplé à un serveur de données distant renseignant sur l'état du réseau d'autobus urbain, l'application s'efforce de guider la personne à travers le réseau lorsqu'elle manifeste le besoin de réaliser l'un des déplacements appris par l'outil. L'orthèse suggère ainsi à la personne de descendre du bus si la destination lui semble mauvaise, et l'aide à trouver l'arrêt adéquat. Malheureusement, les communications réalisées sur ce travail se limitent aux aspects théoriques (voir notamment [118]), aucune évaluation n'ayant été réalisée avec la clientèle.

De son côté, MAPS s'intéresse à la relation avec les tiers, avisant un aidant si le client, en déplacement, semble perdu [29,30]. La communication avec le tiers se concrétise par un message texte envoyé sur son téléphone portable. La compréhension du CAC correspond à un calcul de distance entre la position courante, perçue grâce au récepteur GPS, et un chemin pré-saisi.

L'orthèse MOBUS, de l'Université de Sherbrooke [124], approfondit l'aspect relationnel en couplant l'application du client avec une application dédiée aux aidants, toutes deux fonctionnant sur smartphone. MOBUS se focalisant notamment sur le rappel d'activités, l'orthèse peut aviser l'aidant lorsque le client valide un rappel ou au contraire néglige de le faire. De son côté, l'aidant peut à distance réaliser une mise-à-jour de l'emploi du temps du client. On notera tout particulièrement que les résultats expérimentaux témoignent de la nécessité et de la difficulté à développer des orthèses facilement personnalisables, chaque client étant clairement unique.

[13] Global Positioning System

En conclusion, on retiendra les faits suivants :

- l'adéquation entre le besoin d'assistance et la technologie peut parfois s'avérer très simple. C'est le cas avec le service Neuropage (dispositif de rappel sur téléavertisseur), dont l'efficacité pour une certaine clientèle a été prouvée. L'exemple des applications grand public pour PDA illustre d'ailleurs qu'un outil à consonnance orthétique se doit d'aller à l'essentiel ;
- néanmoins, malgré leur trop grande complexité, même les solutions technologiques grand public donnent des résultats. L'utilisation de la technologie à des fins d'assistance cognitive est une formule sensée ;
- lorsque les besoins sont plus complexes, comme l'aide à la réorganisation du quotidien ou l'aide aux déplacements, les orthèses évoquées illustrent le besoin :
  - de sophistiquer les techniques liées à la compréhension et à l'utilisation du CAC. On pense notamment aux techniques de planification mises en oeuvre par PEAT et MEMOS pour assister l'utilisateur dans sa réorganisation du quotidien ;
  - d'enrichir les modèles de données sous-jacents. Il en est ainsi pour le modèle d'activité de PEAT, offrant la sémantique nécessaire pour permettre une replanification sensée des activités à réaliser ;
  - d'offrir à l'orthèse le moyen de percevoir les informations utiles à l'assistance de façon implicite, comme l'illustre la présence d'un récepteur GPS pour les outils dédiés à la gestion du contexte spatial (en extérieur).

Si l'on se focalise sur les applications pour PDA, la principale limite de ces outils portables semble donc se situer du côté de la perception des données contextuelles, la technologie ne permettant qu'une interaction explicite *via* l'écran tactile et éventuellement le simili-clavier. Cette problématique peut être parfaitement illustrée par le Visual Assistance d'AbleLink [42]. Comme cela a été dit, cette orthèse fournit une aide procédurale, présentant étape par étape la réalisation d'une activité pré-définie. Or la validation d'une étape repose uniquement sur une interaction explicite, soit le fait pour l'utilisateur de sélectionner le bouton "suivant" à l'écran du PDA. En somme, il est parfaitement possible de valider une étape sans l'avoir effectivement réalisée, l'orthèse n'ayant aucune perception des faits et gestes de l'utilisateur sur les objets matériels reliés à l'activité. Même si l'utilisateur est de bonne foi, l'effectivité et la sécurité de la réalisation ne sont en rien garanties. Et la solution technologique ne peut alors se résumer en un simple récepteur GPS portable.

La prochaine section décrit l'apport du paradigme ubiquitaire de l'informatique à la

relation technologie/assistance cognitive. Cette approche met un point d'honneur à repenser les relations entre l'homme et la machine, notamment en favorisant les interactions implicites. Une définition de ce paradigme sera donnée dans un premier temps, puis son application à l'assistance cognitive sera discutée.

2.4 Informatique ubiquitaire et assistance cognitive

2.4.1 Informatique ubiquitaire

Tout a commencé il y a bien longtemps, tout du moins à l'échelle du développement technologique, quelque part en Californie... Et la vision exprimée là-bas au début des années 90 continue de représenter une approche, on ne peut plus moderne, du rapport entre l'homme et la machine.

L'esprit ubiquitaire

En 1991, paraît dans *Scientific American* un article intitulé *The Computer for the 21st Century*, signé par M. Weiser et mettant en vedette l'esprit PARC[14] [157]. Et on peut affirmer que cette approche, qui n'en est finalement qu'à ses premiers pas, a encore de beaux jours devant elle.

L'informatique ubiquitaire[15], c'est avant-tout la volonté de remettre à plat la relation entre la machine et l'homme. Pourquoi l'ordinateur est-il là ? Pour puiser toute notre énergie ou au contraire pour nous permettre de décupler nos capacités ? Pour Weiser, l'"ordinateur personnel", la chose grise avec son écran, son dispositif de pointage et son unité de calcul et de stockage, devant lequel l'utilisateur doit s'incliner pour obtenir l'information qu'il contient, doit disparaître. Plus précisément il doit passer à l'arrière-plan, son rôle étant dorénavant de fournir à l'homme les services nécessaires pour qu'il puisse accomplir ses tâches quotidiennes. L'homme reste alors seul au centre de la scène, libre de focaliser à sa guise son esprit. Et pour cela, il faut arriver à placer judicieusement, dans notre milieu de vie, un peu d'informatique, là où nécessaire. Ces différents éléments, inter-communicants, formeront une communauté de dispositifs au service de la personne. Ainsi l'informatique deviendra ubiquitaire.

[14]Palo Alto Research Center. Centre de recherche fondé par Xerox en 1970, situé en Palo Alto, en Californie. www.parc.com

[15]En anglais, *ubiquitous computing*.

L'informatique ubiquitaire aujourd'hui

L'informatique d'aujourd'hui tend, dans une certaine mesure, vers l'ubiquitaire. En effet, de plus en plus d'outils de traitement automatique de l'information sont disséminés dans notre quotidien, tout en restant à l'arrière plan. Lorsqu'il est au volant de son véhicule équipé d'un système anti-blocage, le conducteur ne se soucie guère du silicone présent dans son automobile : il ne conduit pas un ordinateur, et pourtant celui-ci lui fournit un service en cas d'imminence de blocage de roues. De même, il est possible de trouver des stylos dit "numériques"[16] qui enregistrent les mouvements du poignet tout au long de l'écriture et permettent la retranscription automatique des écrits dans un format numérique. Bien qu'il faille connecter le stylo à un ordinateur personnel pour effectuer la conversion, la préoccupation première de la personne reste l'écriture et non le contrôle d'une machine. Sur le plan de la communication, les progrès en matière de technologie sans-fil facilitent la mise en place d'une communauté d'appareils dans l'environnement. Enfin, il est clair que la miniaturisation des briques technologiques (unités de calcul, de stockage, *etc.*) et l'augmentation de leur capacité favorisent le développement ubiquitaire. Néanmoins, les sauts à plus grande échelle de cette nature restent encore très limités. Mais nombreux sont ceux qui travaillent en ce sens, peut être tout simplement parce qu'il s'agit bien là de la façon la plus naturelle, pour un être humain, de partager son environnement avec des machines.

D'un point de vue conceptuel, ce paradigme s'est posé, construit, enrichi. La maturation acquise depuis le début des années 90 permet aujourd'hui d'en poser plus clairement les principaux aspects et enjeux.

Informatique ubiquitaire ou diffuse ? Si l'expression *ubiquitous computing* peut être traduite par informatique ubiquitaire ou omniprésente, la question de la synonymie entre ces termes et celui d'informatique diffuse[17] se pose. Souvent utilisés de manière indifférente, K. Lyytinen et Y. Yoo apportent la nuance suivante : l'informatique diffuse implique la capacité, pour un ordinateur, d'obtenir et d'utiliser dynamiquement les informations de l'environnement dans lequel il est intégré. L'informatique ubiquitaire ajoute à l'informatique diffuse la mobilité des services (on parle alors d'informatique mobile), qui suivront l'utilisateur au gré de ses déplacements [97].

Un autre terme est présent dans la littérature du domaine, celui d'intelligence am-

[16]Voir par exemple le io2 Digital Pen de Logitech. www.logitech.com

[17]De l'anglais *pervasive computing*

biante, ou *AmI* (pour *Ambient Intelligence*). L'AmI résulte des travaux de l'ISTAG (Information Society Technologies Advisory Group), organisme scientifique de consultation et d'orientation officiant sous l'égide de la Commission Européenne [65]. Si l'AmI s'inscrit pleinement dans le courant ubiquitaire, l'accent est plus particulièrement mis sur le caractère *social* des interactions entre l'homme et la machine, privilégiant la voix et la gestuelle pour communiquer avec les objets pourvus de technologie ayant pris place dans l'environnement [3, 138].

Sensibilité au contexte L'adaptation à l'environnement, aux besoins de la personne, à ses déplacements, sont des objectifs à atteindre pour développer une informatique ubiquitaire. Pour ce faire, il devient nécessaire d'enrichir l'interaction entre l'homme et la machine [45]. Dans l'approche traditionnelle clavier/souris/écran, les échanges ne sont que purement explicites. On perd ainsi ce qui fait la richesse des interactions entre humains, où le non-dit (i.e. tout ce qui n'est pas explicite voire ce qui n'a pas besoin de l'être) constitue une source non négligeable d'informations au sein de l'échange. Ainsi, l'idée est d'élargir le champ de l'interaction homme-machine pour y inclure les informations *implicites*. Cette nouvelle entrée de données devrait permettre à la machine d'avoir une vue plus globale du *contexte* de l'interaction [48]. Le terme consacré pour cette problématique est celui de *sensibilité au contexte*[18].

Plus précisément, A. Dey définit le *contexte* comme "toute information susceptible d'être utilisée pour caractériser la situation d'une entité. Une entité étant une personne, un lieu ou un objet concerné par l'interaction entre une personne et une application, en incluant la personne et l'application" [44]. Cette définition est aujourd'hui largement utilisée dans la littérature, essentiellement parce qu'elle ne fait pas d'*a-priori* sur ce qui caractérise une situation particulière, approche trop coercitive pour être appliquée à l'informatique ubiquitaire. Néanmoins, J. Coutaz reproche à cette définition de ne pas mettre l'utilisateur au centre du contexte, et surtout de ne pas considérer la composition au cours du temps d'états comme éléments contextuels, les observations effectuées précédemment pouvant influencer l'interaction courante. Cette auteur définit le contexte *d'interaction* comme la composition au cours du temps des situations liées à l'utilisateur pour la réalisation de sa tâche[19] [40].

Pour Dey, une application est dite sensible au contexte si "elle utilise le contexte

[18]De l'anglais *context-awareness*. On parle aussi de *conscience du contexte*.

[19]On se reportera au texte de l'auteur pour la définition complète du contexte *d'interaction*.

pour fournir des informations et/ou des services pertinents à l'utilisateur, leur pertinence découlant de la tâche de l'utilisateur" [45]. L'utilisateur et sa démarche, explicite ou implicite, volontaire ou nécessaire (la tâche qu'il complète, souhaite ou doit compléter), prennent sereinement leur place sur le devant de la scène.

2.4.2 Application à l'assistance cognitive

La section 2.3.2, présentant diverses orthèses mobiles, concluait sur leur incapacité à aborder certaines problématiques d'assistance, notamment de par leur capacité très limitée de perception du contexte. L'approche ubiquitaire a alors été entrevue par plusieurs chercheurs comme une alternative intéressante. En voici les justifications.

Lorsqu'ils exposent les fondations de leur système d'assistance cognitive, H. Kautz *et al.* voient dans l'informatique ubiquitaire les ressources nécessaires pour faire de leur système un *partenaire proactif*, capable d'offrir à la personne avec trouble cognitif un support dans l'accomplissement de son quotidien d'une façon aussi naturelle que le ferait un aidant naturel [79, 117]. Grâce à la prise en compte de l'environnement et des informations de localisation de la personne, l'assistance sera fournie à l'intérieur (projet *ADL Prompter*) et à l'extérieur (projet *Activity Compass*, aujourd'hui *Opportunity Knocks*). A l'heure actuelle, le projet d'assistance en intérieur n'a pas été concrétisé, malgré un imposant travail théorique sur la reconnaissance d'activités [116, 120]. Du côté assistance en extérieur, les aspects théoriques (apprentissage de trajet, aide au déplacement dans un réseau de transport public) ont eux aussi été couverts, proposant de nouvelles solutions pour la compréhension et l'utilisation du contexte [118], comme précisé à la section 2.3.2.

A. Mihailidis insiste pour sa part sur les capacités de l'informatique ubiquitaire à fournir une assistance judicieuse [104, 105]. On passerait ainsi d'une diffusion non contrôlée des messages d'assistance à une diffusion contextualisée reposant notamment sur une approche implicite de l'interaction homme-machine. On éviterait ainsi l'émission de messages non nécessaires, inappropriés à la situation voire ineffectifs (émission du message spatialement inadéquate, média inadapté au profil de la personne). En soit l'idée est de se rapprocher des capacités des aidants naturels mises en avant par Kautz. Ces réflexions ont conduit au développement de COACH, une application dédiée à l'assistance au lavage des mains. Très ciblée et relativement intrusive car utilisant comme source d'interaction implicite une vision à base de caméra, cette orthèse a fait l'objet d'une étude de 60 jours en 2004 et présente des résultats intéressants [103]. Testée auprès de dix personnes

présentant une démence modérée à sévère, l'étude montre que le nombre d'étapes que les personnes sont capables de réaliser sans l'assistance d'un tiers augmente de 25% lorsque l'orthèse est présente.

Pervasive healthcare

Plus que celui de l'autonomie fonctionnelle, le domaine de la santé est tout particulièrement ciblé par les approches ubiquitaires. Le premier point abordé concerne le dossier médical informatisé, avec une gestion contextuelle de son contenu [11]. L'idée est ici de moduler, grâce à l'identification de l'interlocuteur et à son rôle communautaire (équipe médicale, usager, famille), l'accessibilité à l'information. Par extension, ces travaux s'appliquent aussi au travail collaboratif des équipes médicales [12, 32]. Mais au delà de ces aspects organisationnels, est née avec l'informatique ubiquitaire une nouvelle approche des soins de santé : le *pervasive healthcare*. Sur le plan technique, I. Korhonen définit cette approche comme "l'application des technologies de l'informatique ubiquitaire aux soins de santé, à la santé et à la gestion du bien-être". D'un point de vue pratique, il la conçoit comme la "possibilité d'accéder en tout temps et tout lieu aux soins de santé" [85]. Les dispositifs de monitorage et de diagnostic sans fil utilisés en institution en sont un exemple. En l'absence d'entraves physiques, l'usager peut dorénavant quitter aisément son lit. Hors milieu institutionnel, on peut citer les dispositifs de *smart medication*, facilitant la gestion des maladies chroniques, où la boîte de médicaments est équipée d'un dispositif technologique ubiquitaire, mobile et répondant au contexte [54]. L'objectif recherché est alors d'aider la personne à gérer sa maladie et à améliorer son acceptation du traitement, en la renseignant sur celui-ci, ses effets secondaires, en effectuant le rappel de la prise voire en détectant l'utilisation de plaquettes de médicaments périmées.

Les techniques ubiquitaires permettent d'envisager à coût abordable les dispositifs de soins personnels [86]. L'aspect économique fait en effet partie des critères de justification : il devient de plus en plus nécessaire, dans nos sociétés, de retarder l'entrée en institution, dont les coûts sont prohibitifs et les conséquences sur les intéressés souvent peu à la hauteur des espérances. C'est le principe du *Aging in Place*, ou comment passer d'une approche "centralisée" des soins à une approche "personnelle", là où vit l'individu [50]. Cette vision se décline en trois facettes technologiques qui composent, en complément des institutions traditionnelles, l'offre globale des soins. La première, le *pervasive healthcare*, vise à promouvoir la santé personnelle et les activités de bien-être, telles qu'illustrées précédemment. La seconde est dédiée à la *télémédecine*, c'est à

dire aux activités médicales visant à promouvoir à distance les rapports entre l'usager et ses intervenants médicaux (par exemple, envoi automatique et régulier de données biomédicales prises au domicile par le biais de dispositifs dédiés aisément manipulables). Enfin la troisième facette vise à faciliter les liens entre la personne et son réseau informel de santé, basé sur les aidants naturels (famille, amis, voisins, *etc.*). L'exemple le plus parlant est certainement celui du *portrait numérique de famille* [132]. Dans ce cadre numérique qui entoure la photographie d'un proche, figurent des icônes reflétant différents aspects liés à sa santé et à son bien-être social. La taille des icônes est automatiquement ajustée en fonction du niveau d'activité, détecté au domicile du proche grâce à un ensemble de capteurs. Un lien ubiquitaire est ainsi constamment tissé entre ces deux foyers, laissant la possibilité aux aidants naturels de se manifester si le niveau d'activité leur semble préoccupant.

Vers un habitat intelligent

L'introduction de dispositifs technologiques à consonance ubiquitaire au sein même de l'espace de vie des personnes fait référence au concept d'*habitat intelligent*. Etabli depuis de nombreuses années, ce concept est, dans le domaine informatique, aujourd'hui indissociable de sa vision ubiquitaire. Néanmoins, cette expression décrit d'une façon plus large l'intégration dans l'habitat de technologies et de services soutenus par un réseau domiciliaire visant à offrir une meilleure qualité de vie à ses occupants[20]. Ainsi la domotique, discipline qui étudie et réalise l'automatisation des installations de l'habitat (chauffage, lumière, gestion de l'énergie, portes et volets automatiques, *etc.*), offre un premier niveau de ressources pour un habitat intelligent. Cela ne correspond toutefois pas à la problématique de ce travail.

A l'instar du laboratoire DOMUS de l'Université de Sherbrooke[21], il existe une petite communauté, grandissante, de centres de recherche travaillant dans cette optique et disposant d'outils adaptés, à savoir d'habitats réels entièrement dédiés à la recherche. "Aware Home" du Georgia Institute of Technology (http ://awarehome.imtc.gatech.edu/), "Gator Tech Smart House" de l'Université de Floride (http ://www.icta.ufl.edu/gt.htm) ou le "House_n / PlaceLab" du MIT (http ://architecture.mit.edu/house_n/) en sont quelques exemples. Les thématiques de ces laboratoires restent toutefois quelque peu éloignées du présent projet. "Aware Home" se focalise plus particulièrement sur le principe du

[20]Définition proposée par www.smart-homes.nl ; consulté le 19 juin 2008.
[21]www.domus.usherbrooke.ca

Aging in Place [2]. On leur doit notamment le *portrait numérique de famille* cité dans la précédente section [132]. L'université de Floride est connue pour son travail sur la couche intergicielle, avec notamment la plate-forme Atlas, destinée à faciliter l'intégration et l'inter-connection de capteurs et d'effecteurs dans un environnement ubiquitaire [66]. Enfin, le laboratoire du MIT, à cheval entre technologie, architecture et design, s'intéresse plus largement à l'inclusion de la technologie dans l'habitat et à son impact sur le comportement humain. Un travail théorique conséquent a toutefois été réalisé concernant la reconnaissance d'activité [145].

2.5 Réflexions conclusives

L'informatique ubiquitaire ouvre de nouveaux horizons quant à l'utilisation de la technologie à des fins d'assistance cognitive. L'intérêt est double : de l'homme vers la machine, l'interaction est ouverte vers l'implicite de par la mise en place dans l'environnement de dispositifs de captation, décuplant les capacités de perception du contexte d'assistance cognitive, et par conséquent la compréhension que peut en avoir l'orthèse. On pense ainsi pouvoir aborder des problématiques comme le suivi de la réalisation d'activités domiciliaires. Mais l'intérêt se trouve aussi dans l'apport d'information, de la machine vers l'homme. La production des actes d'assistance ne sera plus limitée à la traditionnelle interface graphique utilisateur présentée sur un écran. C'est toute l'utilisation du CAC qui peut être repensée, afin de tendre vers une production d'actes plus judicieux.

Dans le prochain chapitre, un scénario illustrera les objectifs de ce travail, quant à l'utilisation de la technologie sous un angle ubiquitaire pour l'assistance cognitive. Avant cela, quelques réflexions conclusives de différentes natures vont être posées. La première rappelle les questionnements éthiques que posent immanquablement une telle utilisation de la technologie. La seconde dresse un court tableau des différentes approches possibles en ce qui concerne la conception des orthèses cognitives. La dernière pose succinctement la question d'une méthodologie de conception basée sur le cadre fonctionnel présenté dans ce chapitre.

2.5.1 Au delà des considérations technologiques

Bien que la technologie soit au centre de ces travaux, c'est bien entendu l'humain qui reste le point de mire de ces recherches. Dès lors, il est naturel que certains problèmes

éthiques se posent. Ainsi, E. Stip et V. Rialle rappellent trois points que les chercheurs se doivent de "surveiller" [140]. Le premier concerne le respect de la vie privée, tout particulièrement dans le cas de la télémédecine, où les données risquent fort d'utiliser les réseaux de communication publics pour transiter de l'habitat à l'aidant professionnel. Le second porte sur les nouvelles responsabilités de la technologie qui, de par la désinstitutionnalisation, se doit d'offrir les services anciennement proposés. Enfin, le troisième point concerne les risques de dépendance envers la technologie, cette question restant complètement ouverte. En somme, c'est toute une nouvelle organisation de l'offre d'accompagnement et de soins qui doit être pensée. Toutefois, au delà de ces aspects normatifs, l'éthique se doit aussi d'être pro-active, en s'interrogeant sur les apports à la clientèle de la technologie face au système traditionnel de santé. Ces questions, dont la pertinence ne peut être mise en cause, ne seront pas réellement abordées dans ce travail. Seule la plus-value technologique offerte par Archipel fera l'objet d'une discussion lors de la présentation des résultats.

2.5.2 Design appliqué à la conception des orthèses cognitives

L'utilisabilité d'une orthèse se pense dès la conception. Ainsi Baecker, comme précisé en introduction, fait du design appliqué à la conception un critère de comparaison des orthèses cognitives. L'approche de base en la matière est la conception centrée-utilisateur, où l'outil est construit en fonction des besoins, capacités et préférences de l'utilisateur. Une méthode traditionnelle est l'interview des clients potentiels [72], mais si un besoin particulier a été ciblé, il est tout aussi possible de s'inspirer des spécificités de ce besoin pour construire l'orthèse. A. Thöne-Otto et K. Walther présentent ainsi une étude pour développer un outil pour le rappel d'activité basée sur un modèle de mémoire prospective [147]. Les différentes phases du processus prospectif sont alors traduites en fonctionnalités logicielles. De même, les capacités technologiques peuvent être mises au centre de la conception - approche centrée-système - pour mieux les ajuster aux demandes des utilisateurs [115]. Mais certains chercheurs préfèrent penser autrement la conception, en ne travaillant plus pour mais avec les utilisateurs. C'est l'approche du design participatif, qui a été exploitée pour l'orthèse OrientingTool, dédiée à l'orientation des personnes avec troubles de mémoire antérograde [159, 161]. Le rôle du client ne se limite alors plus à répondre à des questions mais à suggérer les éléments de conception. Comme on le verra au chapitre 8, certains aspects visuels d'Archipel sont le fruit d'échanges avec des inter-

venants en réadaptation. Toutefois, d'une manière générale, Archipel a été conçu selon une conception centrée-utilisateur.

2.5.3 CAC et méthodologie

Ce chapitre a permis la définition d'un cadre fonctionnel pour les orthèses cognitives, dont certains éléments ont été illustrés grâce aux orthèses existantes. Maintenant établi, il serait intéressant d'introduire une méthodologie de développement utilisant le CAC et les besoins en sensibilité comme critères d'analyse pour la conception d'orthèses cognitives. L'étape suivante serait alors la mise en place d'outils de conception adaptés, les patrons introduits par Maciuszek et cités dans ce chapitre en constituant une première brique significative.

Il serait exagéré d'affirmer qu'un telle méthode d'analyse a été appliquée pour Archipel, l'ensemble de ces concepts ayant pris forme tout au long de la thèse. Bien entendu, les développements présentés ici rentrent pleinement dans ce cadre, avec dans le prochain chapitre une illustration des objectifs de cet outil puis dans les suivants la présentation de la mise en oeuvre réalisée pour la représentation, la perception, la compréhension et l'utilisation du CAC.

Chapitre 3

Archipel : un *framework* objet pour une approche ubiquitaire de l'assistance cognitive

Le choix d'"Archipel" comme nom pour le prototype développé au cours de ce travail n'est pas anodin. Il s'agit en fait d'un pseudo-acronyme (**archi**tecture, **p**romouvoir, autonomie fonctionn**elle**) qui résume grossièrement mais assez fidèlement les objectifs de l'outil et plus globalement de la thèse. Ce chapitre a pour mission de poser ces objectifs. Plus précisément, on exprime ici la vision retenue pour les différents aspects fonctionnels de cette orthèse, dans l'esprit du cadre proposé au chapitre 2 et dans la lignée des besoins exprimés au chapitre 1.

Afin d'améliorer la lisibilité de ces objectifs, ceux-ci sont illustrés et analysés grâce à un scénario d'assistance cognitive, qui sera repris lors de la présentation technique des différentes facettes de l'orthèse. Dans ce chapitre, on discutera aussi diverses considérations méthodologiques, notamment les choix liés à la mise en oeuvre du prototype. On clôturera ainsi la partie introductive de cette thèse.

3.1 Scénarisation et analyse des objectifs

3.1.1 *Mise en situation :* le contexte humain

Pierre Bouchard[1] *est un jeune homme dans la trentaine. A cause d'une malformation génétique, le développement neurologique de Pierre n'a pu se réaliser normalement et il présente une déficience intellectuelle légère. Pierre, qui vit dans son propre appartement, organise au mieux son quotidien avec l'aide de son intervenante. Une activité problématique pour lui est la préparation des repas, qui se traduit le plus souvent pas des plats surgelés réchauffés au micro-onde. Aujourd'hui, Pierre souhaite pouvoir réaliser de façon autonome un ensemble de recettes à son goût. Les cours de cuisine dispensés par le centre où il réalise diverses activités l'ont aidé dans ce sens. Néanmoins, tout le monde s'accorde pour dire qu'il manque à Pierre certaines habilités pour pouvoir réaliser en toute sécurité ce type d'activité. Pierre n'a pas besoin qu'une personne lui dicte la conduite à adopter. Ce qu'il lui faut, c'est un compagnon discret, capable de le guider à sa demande, mais aussi d'intervenir efficacement s'il réalise une erreur manifeste et potentiellement dangereuse.*

L'objectif premier de ce travail est de démontrer la capacité de la technologie à offrir, aux personnes avec troubles cognitifs pour lesquelles un besoin existe, l'assistance nécessaire à la réalisation d'AVQ domiciliaires complexes. Des activités telles que la préparation des repas, qui demandent jugement et planification des tâches, sont à la base d'une vie autonome. Comme présentées précédemment, les études réalisées avec des dispositifs mobiles montrent les capacités de la technologie en matière d'assistance cognitive, tout particulièrement pour les problèmes mnésiques et la planification d'activités simples. Néanmoins, ces orthèses n'ont qu'une sensibilité limitée au contexte d'assistance cognitive. L'analyse qu'elles peuvent faire de la situation est restreinte à celle des interactions explicites de la personne avec l'outil, ce qui ne convient pas au suivi d'activités complexes, potentiellement dangereuses si mal menées. A côté de cela, les chercheurs expriment une grande motivation pour la mise en place de dispositifs ubiquitaires, dont le potentiel en terme d'interaction homme-machine est décuplé. Déjà les récepteurs GPS utilisés pour l'aide à la mobilité transforment les capacités des outils. Toutefois, en ce qui concerne la réalisation d'activités domiciliaires complexes, les travaux concrets restent très limités. Ainsi, mis à part l'assistance pour le lavage des mains, la littérature ne fait

[1] Personnage fictif.

pas écho de résultats en ce sens.

Dès lors, étant donné le cadre fonctionnel présenté au chapitre 2, un objectif plus précis de ce travail est de démontrer la faisabilité d'une application d'inspiration ubiquitaire offrant la sensibilité au contexte d'assistance cognitive nécessaire pour appréhender l'assistance à la réalisation d'AVQ domiciliaires complexes. Les sous-objectifs concernent la représentation des connaissances, la perception, la compréhension et l'utilisation du CAC, ainsi que l'intégration de ces composantes au sein d'une boucle générale de contrôle.

3.1.2 *Mise en situation :* le contexte matériel et technologique

Grâce au partenariat du centre où travaille son intervenante avec un laboratoire de recherche, Pierre peut aujourd'hui bénéficier d'une orthèse cognitive, installée dans son appartement. Concrètement, cette orthèse se présente pour Pierre sous la forme d'un écran tactile connecté à un ordinateur, placé dans sa cuisine, à côté du plan de travail. Néanmoins, il sait que son logement a été équipé de divers dispositifs, permettant à l'ensemble de l'accompagner lors de la réalisation de certaines activités.

Les activités de cuisine, prises comme exemple d'activités domiciliaires complexes, incluent la manipulation d'un grand nombre d'objets et d'appareils. Et ces artefacts jouent un rôle précis dans le contexte de réalisation de l'activité, même si certaines équivalences peuvent être trouvées : un verre, une tasse ou encore un bol peuvent, dans certaines circonstances, offrir une même fonction de contenant à boisson.

Représentation des connaissances

L'objectif en terme de représentation des connaissances est double. Le premier aspect concerne la représentation d'un contexte matériel potentiellement riche : un grand nombre d'artefacts répondant à une certaine classification fonctionnelle. Le second est celui de la modélisation des activités de la vie quotidienne, basée le postulat suivant : cette représentation se définit, concrètement, par la manipulation plus ou moins contrainte d'artefacts matériels. Elle est donc intimement liée à l'aspect matériel.

L'objectif côté compréhension du CAC est le suivi de l'activité, à des fins de détection des erreurs de réalisation (voir *scène 2*). Pour réaliser ce suivi, l'idée est que l'orthèse se base sur la modélisation pré-citée des AVQ, bâtie de façon *a priori* et illustrant la ou les bonnes façons de réaliser l'activité. En témoignant dans la représentation de la richesse du

contexte matériel, on souhaite offrir au processus de compréhension les outils nécessaires à une interprétation souple et sensée des réalisations humaines. Ainsi, l'utilisation d'un verre, d'une tasse ou d'un bol lorsqu'un contenant de ce type est requis pourra donner lieu à une interprétation similaire.

Perception du CAC

Cette logique d'interprétation repose sur la capacité du système à représenter les objets matériels et leurs abstractions, ainsi qu'à transposer ce lien sous l'angle de la perception, la manipulation concernant l'objet concret et non son abstraction.

Les objectifs concernant la perception du CAC sont multiples. Tout d'abord, se pose un problème purement technologique, lié à la multitude des objets matériels dont on souhaite suivre l'utilisation. Il s'agit de déterminer dans quelle mesure on peut et on doit équiper tous ces artefacts, la possibilité technologique pouvant être de plus freinée par des contraintes économiques. Ne pas les équiper signifie faire évoluer l'orthèse dans un univers partiellement observable, tout du moins dans un approche implicite des interactions homme-machine. En effet, il reste parfaitement possible d'interagir explicitement avec le client afin qu'il valide certains aspects que le système ne peut percevoir. L'idée est donc de voir quel compromis peut être réalisé entre l'implicite et l'explicite pour offrir une perception suffisante au suivi de la réalisation, sans prendre pour acquis que chaque geste de la personne pourra être détecté.

Un autre objectif lié à la perception est celui de la représentation des dispositifs technologiques utilisés. Il faut de plus établir la chaîne de raisonnement qui en découle pour l'interprétation des actions atomiques de l'utilisateur, soit faire le lien entre un capteur, l'objet dont la manipulation est détectée par le capteur, et l'action particulière découlant des données du capteur.

Le modèle envisagé pour la représentation des connaissances, qu'ils s'agissent d'objets matériels ou technologiques, comme les capteurs, est présenté au chapitre 4. L'utilisation de ce modèle pour l'interprétation des actions réalisées par la personne est couvert par le chapitre 5, qui s'intéresse aux interactions homme-machine. La représentation des AVQ, qui n'utilise que partiellement le modèle du chapitre 4, est plus précisément illustrée au chapitre 6, qui porte sur le problème du suivi de la réalisation de ces activités.

3.1.3 *Scène 1 :* sortir les ustensiles

A l'heure de la préparation du repas, Pierre clique à l'écran sur "Mes recettes". L'orthèse affiche alors la liste des recettes que Pierre a choisie avec son intervenante, en fonction de leur degré de difficulté et des goûts de Pierre. Pour aujourd'hui, il sélectionne la recette de spaghettis à la bolognaise. Le système en présente alors la première étape. Il s'agit de sortir tous les ustensiles et de les mettre sur le plan de travail. Cette approche très systématique rappelle les cours de cuisine suivis par Pierre au centre. Bien que parfois contraignante, il sait que cette méthode permet de structurer la préparation du repas, et c'est exactement ce dont il a besoin. Pierre n'éprouve aucune difficulté à sortir les ustensiles usuels. Pour les cuillères à mesurer, il demande au système de l'aider à les retrouver. L'orthèse lui répond par un petit signal sonore d'acquiescement et met en évidence un des tiroirs de la cuisine, grâce à un système d'éclairage. Une fois le tiroir ouvert, et les cuillères sorties, la lumière s'éteint. Pierre vérifie alors à l'écran s'il dispose de tous les ustensiles. Il se base sur les vignettes qui représentent chacun des ustensiles. C'est d'ailleurs en appuyant sur la vignette des cuillères que le système l'a précédemment aidé.

Cette scène illustre plusieurs actes d'assistances produits à la demande de l'utilisateur, notamment une aide procédurale (en réponse à la sélection de l'activité) et une aide mnésique (localisation d'un objet dans l'environnement). Sur le plan de l'utilisation du CAC à des fins d'assistance, l'objectif est de fournir l'infrastructure nécessaire pour permettre le développement d'outils d'assistance spécifiques, tels que ceux illustrés ici.

Sans rentrer dans les détails de ces assistants particuliers, on peut dresser le portrait global de la mécanique d'interaction envisagée. Globalement, l'acte d'assistance est ici vu comme un message que l'on souhaite adresser à la personne assistée. En fonction du besoin d'assistance, la nature du message va être différente. Dans le cas de la localisation d'un objet, l'idée est de mettre en évidence l'objet dans l'environnement, l'assistance étant projetée au coeur même de la réalisation de l'activité, dans un esprit ubiquitaire des interactions machine-homme.

C'est tout d'abord le contexte spatial qui va être impliqué, puisqu'il faut déterminer où l'information doit être diffusée. En l'occurrence, la question est de savoir où trouver les cuillères à mesurer. Dans un second temps, l'idée est de recouper contexte spatial et technologique, afin de découvrir quel dispositif peut être utilisé pour transmettre le type de message à l'emplacement souhaité. Enfin, il faut pouvoir mettre en oeuvre le

dispositif pour rendre effective l'interaction, et éventuellement s'assurer que le message a bien été reçu. Concrètement, ces différents aspects seront essentiellement abordés sous l'angle de la représentation des connaissances et des raisonnements à réaliser autour de ces connaissances, présentés au chapitre 5.

3.1.4 *Scène 2 :* sortir les ingrédients

Comme tous les ustensiles sont sortis, Pierre appuie sur "étape suivante" et doit maintenant faire de même pour les ingrédients. Sûr de lui, il sort ce qu'il faut puis appuie à nouveau sur "étape suivante". Le système diffuse alors un message audio demandant à Pierre de vérifier s'il a véritablement sorti tous les ingrédients. Après vérification grâce aux vignettes, il s'aperçoit qu'il a oublié les épices.

Dans certaines situations, les actes d'assistance doivent être produits de façon automatique, soit non plus à la demande du client mais en réponse à une erreur commise au cours de la réalisation d'une activité, dont le suivi est assuré par l'orthèse. Pour l'orthèse, l'idée est de voir une erreur comme la non-validation d'une contrainte de réalisation, ces contraintes étant exprimées par le modèle de l'activité. En l'occurrence, l'action de sortir les épices n'a pas été réalisée avant la validation explicite par Pierre de la fin de l'étape[2]. La séquence de réalisation n'est donc pas conforme au modèle et plus précisément à sa contrainte d'ordonnancement.

Un des objectifs, du point de vue de la compréhension du CAC, est non seulement de détecter les erreurs de réalisation mais aussi de leur donner une sémantique, i.e. de les *qualifier*. Lors de la présentation du cadre fonctionnel, la notion de profil d'assistance a été introduite. Par profil d'assistance, on entend la description de l'approche d'assistance à mettre en oeuvre lorsque telle problématique cognitive est rencontrée. L'objectif de la qualification des erreurs est de tenter d'établir un parallèle entre la détection algorithmique d'une erreur (non-validation d'une contrainte de réalisation) et certaines problématiques cognitives, afin de mettre en oeuvre l'assistance adaptée, telle que précisée par le profil. Dans cette scène, l'erreur d'ordonnancement pourrait être rapprochée d'un problème exécutif de planification, auquel l'orthèse répond en diffusant un message incitant à vérifier la bonne réalisation de l'étape. La qualification des erreurs sera abordée

[2]Le chapitre 7 détaille comment, à partir du modèle d'une activité, une procédure de réalisation peut être construite et présentée à l'utilisateur sous forme d'étapes dont certaines nécessiteront une confirmation explicite.

au chapitre 6, le profil d'assistance au chapitre 7.

3.1.5 *Scène 3 :* la préparation des légumes

L'étape suivante est la préparation des légumes. Perfectionniste, Pierre profite à chaque fois des vidéos qui lui sont proposées pour s'assurer qu'il coupe comme il se doit les légumes. Un chef cuisinier a bien voulu participer à la réalisation des vidéos illustratives. Une fois les légumes coupés, Pierre passe à l'étape de cuisson. Tout se déroule à merveille lorsque le téléphone sonne.

En s'inspirant du paradigme ubiquitaire, on souhaite multiplier les possibilités d'interaction homme-machine, dans l'espace et dans les modalités. L'objectif est clair : améliorer la compréhension voire l'acceptabilité du message d'assistance. Dans la présente scène, l'interaction est assez traditionnelle, avec un contenu vidéo explicatif diffusé via l'écran tactile. Pour la première scène, le système envisagé est plus novateur, un éclairage diffus servant de support pour la transmission de l'information.

L'innovation ne se trouve toutefois pas dans le support, mais plutôt dans l'utilisation qui en est faîte. Le design de nouveaux supports d'interaction ne fait pas partie des objectifs de ce travail. L'idée est plutôt de réfléchir à la façon de gérer une interaction dans un environnement ubiquitaire, en l'illustrant grâce à des dispositifs existants, ainsi que de penser à la manière dont l'acte d'assistance doit être construit en amont. Les chapitres 5 (interaction) et 7 (assistance) abordent respectivement ces deux aspects.

3.1.6 *Scène 4 :* le temps de cuisson

Pierre va alors répondre au téléphone. Il s'assoit dans un fauteuil pour discuter et c'est en raccrochant que son attention est captée par un magazine sur les timbres, dont Pierre est collectionneur. Au bout de quelques minutes, un message audio lui rappelle que le temps de cuisson est écoulé et que la cuisinière est toujours allumée. Pierre, absorbé par le magazine, avait en effet oublié la préparation en cours. Il se rend en cuisine et éteint la cuisinière. Puis il termine correctement la recette, fait la vaisselle et remet chaque ustensile à sa place, l'orthèse s'en assurant.

Les activités prenant place dans la cuisine constitueront, dans ce travail, l'exemple type d'activités domiciliaires complexes. Néanmoins, le prototype réalisé n'est en rien limité à l'aide à la réalisation de recettes. De plus, cette scène illustre que le suivi de

la réalisation d'une activité prenant place en cuisine n'est pas forcément limitée aux frontières de cette pièce, puisque le message d'assistance doit pouvoir être perçu par Pierre, alors installé dans son salon.

Dans cet exemple se pose le problème du suivi des déplacements de l'utilisateur. Divers travaux ont été réalisés autour de cette thématique (par ex., [129]), offrant diverses solutions qu'il serait certainement pertinent de mettre en oeuvre au sein de la présente application. Et cette remarque s'applique à nombre de problématiques.

Dès lors, l'idée est d'offrir le cadre applicatif nécessaire et suffisant pour la perception, la compréhension et l'utilisation du CAC à des fins d'assistance à la réalisation d'activités domiciliaires complexes, sans préjuger de certains choix spécifiques de mise en oeuvre. On pense par exemple à la mécanique de validation des contraintes pour la détection des erreurs. Néanmoins, une solution sera proposée et implémenter pour ces différents aspects, afin de faire de cette application un prototype d'orthèse viable et expérimentable.

3.2 Méthode et choix de mise en oeuvre

Ce scénario a permis d'illustrer et par la même de préciser les objectifs de ce travail. Dans cette section, certaines considérations méthodologiques et portant sur la mise en oeuvre sont précisées. Il est tout d'abord question des limitations portées à la problématique afin de garantir la faisabilité de ce projet dans le cadre d'une thèse, puis la notion de *framework* et son application au présent travail sont précisées. Enfin, on spécifie en quoi le domaine d'application qu'est l'assistance cognitive influence la façon de programmer.

3.2.1 Limitations apportées à la problématique

Pour faire de cette problématique un travail de thèse réaliste, les limitations suivantes ont été apportées. Tout d'abord, bien que l'informatique ubiquitaire suppose une répartition des traitements liés ici à la perception, compréhension et utilisation du CAC à des fins d'assistance, ils resteront centralisés. Bien entendu, il y aura répartition dans l'environnement de capteurs et d'effecteurs, mais ceux-ci seront dénués d'intelligence locale, et se limiteront au relais d'informations. Ensuite, la démarche d'assistance ne portera que sur une seule personne à la fois. La problématique d'identification des personnes ne sera donc pas abordée, et la compréhension des actions supposera l'uniformité de leur origine (toute action perçue est supposée liée à une même et unique personne). Enfin,

la personne assistée ne réalisera qu'une activité à la fois. A noter que si cette dernière remarque est d'une manière générale irréaliste, elle a un certain fondement pour les personnes souffrant de troubles cognitifs. En effet, il peut être souhaitable, pour faciliter la réussite des activités, que la personne n'en réalise qu'une seule à la fois.

3.2.2 Un *framework* objet

Si Archipel se présente aujourd'hui sous la forme d'un *framework*, ce n'est pas par volonté d'atteindre un objectif initial. C'est par contre l'aboutissement d'une longue réflexion sur ce que doit être, au final, une telle application. F. Pachet définit les *frameworks*[3] comme des "squelettes d'application qui imposent une conception figée pour une classe d'applications donnée, tout en donnant la possibilité de redéfinir certaines parties du squelette existant. Le plus souvent ces *frameworks* imposent en fait une boucle de contrôle générale dont les parties difficiles sont déjà codées, sous la forme le plus souvent de classes abstraites, et qui appelle des "bouts de code" dont l'écriture est laissée aux soins de l'utilisateur, le plus souvent sous la forme de sous-classes concrètes." [110]. Dans ce travail, la classe d'applications est celle des applications d'assistance cognitive à forte sensibilité au CAC. Leur boucle de contrôle générale suit une logique de perception, compréhension, puis utilisation du CAC. Côté "utilisateur", la littérature fait état de diverses approches algorithmiques pour la compréhension, et il existe un grand nombre de stratégies de génération et de délivrance des actes d'assistance. L'ensemble s'applique parfaitement à la mise en place d'un *framework*.

Le choix d'un langage objet pour le développement d'un *framework* est une approche assez traditionnelle, l'esprit *framework* s'alliant parfaitement à la conception par objets. Pour Archipel, ce langage sera Java. La justification réside notamment dans les orientations du laboratoire DOMUS, la portabilité du langage répondant à la non uniformité des supports d'exécution dans un environnement ubiquitaire. C'est à nouveau un choix tout à fait commun pour le domaine.

Dans son "livre blanc" sur la conception de *frameworks* objets [71], Taligent différencie plusieurs types de *frameworks*. On trouve notamment les *frameworks* applicatifs, qui encapsulent une expertise applicable à une large variété de programmes, ainsi que les *frameworks* de domaine, pour lesquels l'expertise ne s'applique qu'à un domaine particulier. La figure 3.1 présente les *frameworks* et librairies du projet Archipel. On re-

[3] "Cadre d'applications" a été proposé comme traduction pour *framework* par l'Office québécois de la langue française. Le terme anglo-saxon lui sera préféré dans ce travail.

trouve les *frameworks* applicatifs archipel.fwk.appmanager, qui permet de gérer le cycle de vie d'une application, domus.xml pour la gestion de la représentation d'objets en XML, et domus.beanbuilder, dédié à l'édition graphique des propriétés des objets. Du côté des librairies, on trouve des modèles de données pour le temps (domus.time), les AVQ (domus.task), la catégorisation d'objets de la vie quotidienne (domus.odlcategorizer) et la représentation des expérimentations (domus.experimentation). Ces composants seront détaillés ultérieurement.

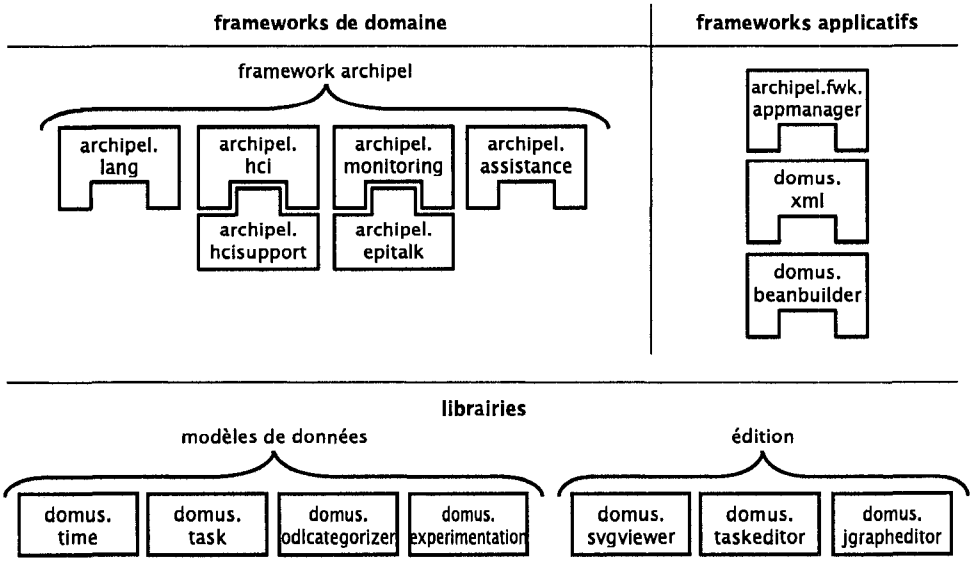

FIGURE 3.1 – *Frameworks* et librairies du projet Archipel

Le coeur d'Archipel se situe au niveau des *frameworks* de domaine. Archipel est en fait découpé en 4 sous-*frameworks*. Le premier, archipel.lang, concerne la représentation des connaissances. Il sera présenté le premier (chapitre 4), tant la représentation du CAC a une place prépondérante dans cette application. Le second, archipel.hci, est bien entendu axé sur l'interaction homme-machine. Il répond à la problématique de perception du CAC, et offre un ensemble de services pour l'utilisation du CAC. Il sera présenté au chapitre 5, ainsi qu'à titre d'étude de cas les classes concrètes d'archipel.hcisupport. Le troisième, qui est présenté au chapitre 6, est archipel.monitoring. Ce sous-*framework* porte sur la compréhension du CAC. Répondant à cette problématique sous l'angle du monitorage d'activités, il sera complété par la présentation d'archipel.epitalk, mise en oeuvre concrète du monitorage basée sur le *système conseiller EpiTalk* [114]. Enfin, archipel.assistance fera

l'objet du chapitre 7. Celui-ci concerne l'utilisation du CAC et plus précisément la notion d'acte d'assistance. A noter que l'application utilise aussi des librairies tierces qui seront citées aux moments opportuns[4].

3.2.3 Une programmation axée sur l'assistance cognitive

Le développement d'applications sensibles au contexte a poussé les chercheurs à s'interroger sur leurs habitudes de programmation. En effet, si de nombreux aspects doivent être abordés au niveau *middleware* (acquisition des données bas niveau issues de capteur, interprétation en données de plus haut niveau, transmission des données interprétées aux applications), la couche applicative n'est pas en reste : le comportement du programme doit être capable de s'adapter aux changements[5]. Ainsi est née la Programmation Orientée Contexte (POC), dont l'objectif est le suivant : éviter au développeur de devoir "répandre" à travers le programme des comportements qui dépendent d'un contexte donné [38].

Les pires scénarios peuvent être envisagés dans le cas contraire, comme de magnifiques structures conditionnelles offrant les alternatives possibles (et exhaustives ?) en fonction des différents contextes que l'on présume rencontrer... Bien entendu, ces solutions ne sont absolument pas viables et contredisent pleinement l'idée même de modularité qui semble indissociable de l'adaptation au contexte.

Bien que l'idée générale reste, la notion de POC est abordée de façon concrète sous des angles bien différents. La première approche est celle des *layers* de P. Costanza et R. Hirschfeld [38]. Dans celle-ci, les classes sont découpées en couches qui peuvent être ou non activées en fonction du contexte, offrant ainsi différentes vues virtuelles sur l'objet. Ainsi, chaque élément de la classe (propriété, méthode) est associé à une couche et prend vie ou disparaît en fonction de son activation. Une autre approche, proposée par R. Keays et A. Rakotonirainy, repose sur l'idée d'un code mixte, contenant des parties traditionnelles indépendantes du contexte et des "termes ouverts", dont le contenu changera en fonction de celui-ci [80]. Aux termes ouverts sont associés des buts, qui identifient le rôle du

[4]Les librairies tierces restent à usage utilitaire. Le coeur de l'application correspond aux composants présentés à la figure 3.1 et représentent - bien que cela n'ait pas de réelle signification en soi - environ 850 classes et interfaces Java.

[5]Dans une approche profondément ubiquitaire de l'informatique, il existe un grand nombre de problématiques autres liées à la continuité des services lorsque les personnes et matériels se déplacent dans l'environnement (migration des applications, adaptation aux capacités de l'infrastructure réseau, tolérance aux fautes, *etc.*) [63].

code manquant, et qui sont utilisés au cours de l'exécution pour aller chercher dans une librairie un "morceau" de code de même but et applicable au contexte d'exécution (défini par divers critères).

En l'absence d'applications "grandeur nature" déployées avec ces approches (on ne parle que preuve de concept), il reste difficile de cerner leur utilisabilité, tout particulièrement pour l'approche par couche. En fait, les préoccupations de ces auteurs semblent un peu loin des préoccupations "de terrain" d'applications spécifiques telles que Archipel. Ainsi, si la POC doit sous-entendre des capacités d'adaptation au contexte, elle doit avant tout mettre l'information contextuelle au coeur de la programmation, problématique à laquelle on tentera de répondre dans le prochain chapitre. On gardera alors à l'esprit que l'information contextuelle présente dans le programme doit rester une abstraction des différents contextes concrets d'exécution, afin de satisfaire le critère d'adaptabilité [31].

Ainsi se terminent les chapitres introductifs de cette thèse, qui ont permis de poser la problématique de l'assistance cognitive, tant du côté des clients que de celui des chercheurs et des développeurs. Le prochain chapitre explique pour sa part la représentation des connaissances (RC) liées au CAC dans Archipel. Grâce à son approche originale, Archipel offre aux connaissances la possibilité d'être manipulées comme des objets de premier ordre, tout en laissant le choix au développeur de la sémantique de cette représentation.

Chapitre 4

archipel.lang

On peut résumer ainsi la définition proposée par R. Davis de la représentation des connaissances (RC) [43], citée au chapitre 2 : la RC est intimement partiale, liée aux besoins de l'application. Elle est un filtre sur le monde réel, n'autorisant que les connaissances jugées utiles pour la description du domaine et les mécanismes de manipulation associés.

La raison d'être d'Archipel est l'aide à la réalisation d'activités domiciliaires complexes. Pour satisfaire cet objectif, on envisage une perception, une compréhension et une utilisation du CAC reposant largement sur les connaissances contextuelles, ainsi que l'illustre le scénario présenté au chapitre 3. Plus précisément, les connaissances contextuelles représentées doivent permettre au système de raisonner en matière d'interactions homme-machine et de suivi de la réalisation des activités.

Ce chapitre présente le langage de représentation de connaissances qui a été mis en oeuvre pour répondre à cette attente. Dans un premier temps, les aspects représentation du scénario sont repris, afin de dresser un portrait global des besoins. Est mise de côté celle des AVQ, qui sera spécifiquement abordée au chapitre 6. Puis le langage de RC d'Archipel est introduit, étape par étape, afin d'en comprendre la logique de conception et ses possibilités d'utilisation. On notera à ce sujet que le chapitre 5, qui porte sur les interactions homme-machine, propose une illustration beaucoup plus complète de l'utilisation de ce langage, le présent chapitre portant essentiellement sur l'explication de la mécanique sous-jacente. Enfin, une discussion sur d'autres possibilités en matière de RC vient clore le chapitre. On présente ainsi pourquoi Archipel va à l'encontre de la tendance actuelle du domaine, très imprégnée de logiques de description et de leur

langage de prédilection : OWL[1].

4.1 Vers un langage de représentation des connaissances par objets

4.1.1 Retour sur le scénario présenté au chapitre 3

L'histoire de la tasse que l'on peut utiliser au même titre que le bol ou le verre lorsqu'il est question d'un contenant à boisson est le premier aspect à considérer. De prime abord, la représentation du contexte matériel est celle d'un ensemble de concepts que l'on doit pouvoir classifier, conformément à leur utilité matérielle pour la réalisation des activités. Ainsi verres, bols et tasses sont des contenants à boisson. Plus précisément, il existe une relation sémantique entre le concept de verre et celui de contenant à boisson, applicable au concept de bol et de tasse. Les concepts de verre, bol et tasse font alors référence à l'ensemble des verres, bols et tasses concrets présents dans l'environnement. Il est donc question de concepts plus ou moins abstraits et d'objets concrets, liés par des relations sémantiques. Cette représentation prend la forme d'une ontologie.

Pour qu'il puisse y avoir perception du CAC, ces objets matériels devront, selon une logique d'interaction implicite, être équipés de capteurs. Les relations s'étendent alors du contexte matériel vers le technologique : tel capteur est associé à tel artefact. Ces relations ne sont toutefois plus sémantiques, mais purement matérielles. Elles sont nécessaires pour mener à bien l'interprétation des données des capteurs, puisque le suivi des activités est basé sur la manipulation des artefacts, et non celle des capteurs. De même, il est indispensable de préciser quelle logique d'interprétation s'applique à tel capteur, voire à telle sorte de capteur. Ainsi, la représentation doit permettre de préciser les caractéristiques de ces entités, celles-ci pouvant être simplement descriptives (le capteur est connecté à tel artefact) ou plus comportementales. Dans ce second cas, il peut par exemple s'agir de préciser quel traitement algorithmique doit être réalisé pour assurer l'interprétation des données physiques du capteur. Si ce traitement s'applique à l'ensemble des capteurs d'un type donné, ce qui semble logique, alors cette caractéristique comportementale sera associée non plus à la représentation d'une entité concrète mais à celle d'un concept abstrait.

[1] *Ontology Web Language*

Dans le scénario, Pierre Bouchard est amené à quitter la cuisine, pièce où se déroule l'activité. Pour le système, cette information se doit d'être connue afin d'assurer la bonne diffusion spatiale des informations d'assistance. Plus précisément, il faut pouvoir associer à Pierre un emplacement, une telle information s'appliquant aussi aux artefacts matériels et technologiques. Cela sous-entend que le contexte humain, représenté ici par Pierre, et le contexte spatial doivent rentrer dans le cercle des connaissances. Par représentation du contexte spatial, on suppose la description des emplacements de l'espace de réalisation des activités, à une échelle utile pour l'assistance. La notion de pièce peut par exemple poser les limites spatiales de réalisation d'une activité. Un découpage de la pièce en zones d'intérêt pourra s'avérer nécessaire pour le suivi de la réalisation. On peut penser aux plans de travail ou la zone de la plaque de cuisson dans la cuisine. La représentation devra aussi témoigner des relations existantes entre ces emplacements (le plan de travail *est* dans la cuisine). Enfin, ce découpage doit rester cohérent avec les possibilités technologiques de localisation.

Lors de la production d'un acte d'assistance, une des problématiques est de trouver un effecteur capable d'offrir l'interaction désirée, tant sur le plan spatial que sur le plan de la sémantique du message. Lorsque Pierre demande au système de l'aider à localiser les cuillères à mesurer, l'idée proposée est de mettre en évidence l'emplacement des cuillères, afin de guider Pierre en attirant son attention. Dans le scénario, cette interaction est réalisée par un dispositif d'éclairage local à cet emplacement, que l'on pourrait faire clignoter. Pour permettre le raisonnement, outre les caractéristiques spatiales des cuillères et du dispositif d'éclairage, il faut décrire les services d'IHM que peut offrir ce dispositif. En l'occurrence, on considère qu'un éclairage permet de mettre en évidence un emplacement. Cette notion de service est un nouveau concept, des plus abstraits, qui doit lui aussi être décrit. Une relation sémantique pourra alors être établie entre l'effecteur et le service.

La production de l'acte se concrétise par sa mise en oeuvre réelle, soit l'allumage du dispositif d'éclairage dans la cas présent. Bien entendu, le contrôle de l'effecteur doit pouvoir se faire de façon logicielle. Mais ce qui importe ici, c'est la façon dont ce contrôle logiciel est rendu possible, à partir de l'entité représentant l'effecteur dans le modèle de connaissance. La représentation doit permettre une manipulation complète et transparente des outils à disposition de l'orthèse pour la diffusion des actes d'assistance.

4.1.2 Principe général de la mécanique de représentation

En complément de ces considérations sur la nature de la connaissance à décrire, un des enjeux de cette représentation est de faire de la connaissance un citoyen de première classe du *framework* Archipel. Autrement dit, les informations contextuelles doivent pouvoir être manipulées au sein de l'application comme tout autre entité, conformément aux principes du langage de programmation d'Archipel, Java.

Parmi les spécificités de Java, on retiendra ici que c'est un langage de programmation *par objets*. Comme le dit si bien J.F. Perrot, le paradigme objet "rapproche les structures informatiques des formes et des intuitions de la pensée ordinaire [...] L'héritage culturel de l'*homo faber* nous donne une efficacité extrême dans la manipulation des objets matériels (le plus souvent vus comme des outils) : c'est cette efficacité qu'on souhaite transférer dans le domaine de la programmation" [119]. Très simplement, un objet est une entité informatique individuelle, définie par son état et son comportement. Un objet possède en effet un ensemble d'attributs, qui le décrivent. Et c'est la valeur prise par ses différents attributs qui donne à l'objet son état. A côté de cela, un objet peut être actionné, en fonction de son comportement. Celui-ci est défini par les méthodes de l'objet, soit un ensemble de procédures pouvant être exécutées dans le contexte local de l'objet. Pour ce faire, il suffit de communiquer avec l'objet, en lui envoyant un message. Si l'objet le comprend, autrement-dit si ce message s'inscrit dans ses limites comportementales, alors l'objet s'actionnera et ce conformément à son état courant. Cela pourra d'ailleurs avoir pour conséquence la modification de cet état.

Cela étant dit, la question est de savoir comment définir les attributs et le comportement d'un objet. La solution la plus commune est celle des *langages à classes*. Dans cette approche du paradigme objet, attributs et comportement sont donnés par la *classe* de l'objet. Ce dernier devient alors une *instance* particulière de cette classe, disposant de ses propres valeurs d'attributs. La classe peut donc être vue comme la représentation d'un concept abstrait, dont l'instance est une réalisation concrète.

Une autre caractéristique du paradigme objet que l'on citera ici est *l'héritage*. Dans les langages à classes, l'héritage se traduit par l'établissement d'une relation d'ordre entre classes d'objets, celle de leur plus ou moins grande généralité. La hiérarchie d'abstraction ainsi établie permet aux classes de partager la définition de leurs attributs et de leur comportement, les classes de plus bas niveau héritant des caractéristiques de leurs supérieures. Cette possibilité est généralement vue comme la caractéristique majeure du paradigme objet [119].

Ces quelques considérations sur le modèle objet ouvrent la porte à une certaine approche de la représentation des concepts contextuels. Les caractéristiques descriptives et comportementales des éléments contextuels semblent pouvoir prendre forme au sein de ce modèle, appuyées par le principe de l'héritage qui établit un certain type de relation sémantique entre concepts. Le paradigme objet constituera la base de cette réflexion sur la représentation des connaissances.

Pour mettre en oeuvre cette RC par objets, le choix le plus simple semble être le langage Java, puisque celui-ci est d'ores et déjà présent au sein de l'application. Malheureusement, cette solution se trouve limitée par les caractéristiques du modèle objet de Java, qui, comme on le verra par la suite, font de Java un langage à classes incomplet. Néanmoins, il semble judicieux de garder Java comme standard de développement, afin d'assurer l'intégration des connaissances contextuelles dans le *framework*. L'idée est donc de créer une sur-couche Java assurant la représentation recherchée sans contredire les bases de ce langage. C'est ce que présente la suite de ce chapitre, sous l'appellation archipel.lang.

Plus précisément, cette sur-couche doit être suffisamment souple pour répondre à l'ensemble des besoins exprimés précédemment. Ces besoins sont de natures diverses, et ce d'autant plus qu'ils ne se limitent pas à la description du contexte : il est aussi question de raisonnement, et de capacité de la représentation à servir d'intermédiaire naturel pour manipuler les entités réelles de l'environnement, comme les effecteurs. Le tout s'inscrit enfin dans un *framework*, soit un outil sensé permettre certaines largesses dans la mise en oeuvre, même si le parcours reste balisé.

En somme, on ne souhaite pas préjuger des besoins réels de chaque représentation. Ainsi, ce n'est pas un langage mais un ensemble de langages de représentation permettant, dans une certaine mesure, le raisonnement sur les connaissances et leur manipulation matérielle qu'archipel.lang doit supporter. Chaque langage permettra alors la création d'un ensemble de modèles objets de connaissances, utilisables au sein du *framework*. C'est à ce type de problème que répondent les protocoles à méta-objets, qui sont introduits dans la prochaine section.

4.2 Notion de protocole à méta-objets

4.2.1 L'art du MOP

La notion de protocole à méta-objects (MOP[2]) a été introduite par G. Kiczales et ses collaborateurs en 1991, et présentée au sein d'un ouvrage au nom évocateur : *The Art of the Metaobject Protocol* [82]. Originellement pensé pour CLOS[3], le concept de MOP s'est avéré applicable à tout langage de haut niveau, de par son objectif relativement universel : permettre à un langage d'être suffisamment extensible pour répondre à l'ensemble des demandes exprimées par les utilisateurs, dans les limites de sa raison d'être. Pour illustrer cette idée, Kiczales cite les problèmes de performance de CLOS, jugée faible par les utilisateurs malgré un pouvoir expressif répondant aux attentes des plus sceptiques [81]. Cela était dû à la façon dont les instances étaient gérées, tout particulièrement en ce qui concerne l'accès aux valeurs des champs. En effet, une structure de type table de hachage s'avère plus efficace qu'un tableau lorsqu'une instance comporte un très grand nombre de champs mais que peu d'entre eux sont utilisés. On limite le parcours coûteux d'une structure largement vide. En offrant la capacité aux programmeurs de modifier la stratégie d'accès aux valeurs en fonction des caractéristiques du domaine, le MOP CLOS alliait expressivité et performance. En somme, le bonheur pour l'utilisateur.

Cela implique toutefois de penser autrement les langages, de passer de la traditionnelle boîte noire à un langage ouvert, où des choix profonds de mise en oeuvre sont laissés à la discrétion de l'utilisateur. Du point de vue du concepteur, cela veut dire élargir sa vision : au lieu d'envisager un langage comme un support pour une gamme finie de programmes (figure 4.1 (a)), il faut entrevoir une plus large étendue de programmes supportés par une gamme de langage, dont le MOP est la base, l'abstraction (figure 4.1 (b)). Du point de vue de la représentation, du raisonnement et de la manipulation des connaissances par objets, l'idée est de proposer un MOP offrant la capacité de créer un panel de langages de ce type dont la sémantique s'accordera aux besoins de différentes représentations (figure 4.1 (c)). Ainsi, les MOPs sont des "interfaces au langage qui offrent aux utilisateurs la possibilité de modifier incrémentalement le comportement et l'implémentation du langage, de manière analogue à la possibilité d'écrire des programmes avec le langage" [82].

[2]Acronyme pour MetaObject Protocol
[3]Common Lisp Object System, langage objet basé sur Lisp.

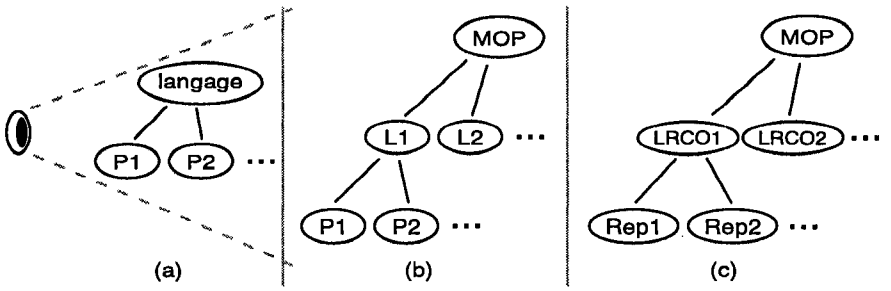

FIGURE 4.1 – Elargissement de la vision du concepteur de langage : du langage fermé au protocole à méta-objets. (a) et (b) repris de [81]

4.2.2 Concepts d'un MOP

Les MOP sont nés suite à l'introduction de la *réflexivité* dans les langages objets. Un système est dit réflexif s'il est capable "d'observer et de modifier la manière dont sont réalisés les traitements qu'il effectue" [133]. L'observation, et plus largement la capacité du système à répondre à des questions concernant son état correspond à une démarche d'*introspection*. Si le système s'auto-modifie, on parle alors d'*intercession*. Introspection et intercession nécessitent que l'état du système soit représenté sous forme de données manipulables par ce même système. Ce processus est appelé *réification* (du latin *res*, chose : on pourrait dire *chosification* [119]), et correspond à la construction de l'auto-représentation d'un système réflexif [133].

Dans sa thèse de doctorat, M. Bouraqadi propose une définition très complète des concepts liés aux systèmes réflexifs en général et aux MOP en particulier [133]. Ces définitions sont reprises ici.

Méta-objet Un *méta-objet* est un "objet qui joue le rôle d'interprète pour un ou plusieurs autres objets. Les méta-objets permettent de contrôler la structure et le comportement d'autres objets. Par exemple, un méta-objet définit la manière d'allouer la mémoire qui correspond à la structure des objets qu'il contrôle" ;

Réification La *réification* est le "résultat de l'opération de même nom qui consiste à représenter des éléments de programmes sous forme d'objets qui existent à l'exécution. A titre d'exemple, les classes et les méthodes constituent des réifications dans un langage réflexif" ;

Objets du méta-niveau "Un méta-niveau est constitué de méta-objets et de réifications. Nous désignons donc par objet du méta-niveau, un objet faisant partie du méta-

niveau sans préciser si cet objet est un méta-objet ou une réification".

Ces définitions permettent à M. Bouraqadi de préciser la notion de MOP, qu'il définit comme "l'interface qui permet de manipuler tous les objets du méta-niveau"[4]. La définition originale de G. Kiczales ne considérait que les réifications (alors définies comme des méta-objets) [133]. M. Bouraqadi l'étend aux méta-objets tels qu'introduits par P. Maes [99], dans l'acception donnée ci-dessus.

La notion de méta-niveau découle de la séparation des services et des mécanismes d'exécution dans les systèmes réflexifs. Au niveau de base se trouvent les services ou "fonctionnalités", i.e. le "Quoi" du système. A une abstraction au dessus, au niveau méta, sont décrits les mécanismes d'exécution du système, i.e. le "Comment". L'ensemble offre une plus grande modularité que dans les systèmes classiques, l'un des niveaux pouvant être modifié de façon indépendante de l'autre. Mais il permet avant tout une plus grande adaptabilité, car il devient possible d'intervenir sur les mécanismes d'exécution pour les conformer aux besoins de l'utilisateur.

4.2.3 Mise en place d'un MOP

Comme le précise M. Bouraqadi, la première chose à considérer est l'identification des réifications et des méta-objets, soient les entités et concepts manipulés au méta-niveau, ainsi que les relations qui les lient. Plus précisément, cela revient à cibler les mécanismes d'exécution contrôlés par les méta-objets et à expliciter les entités représentant les programmes. A noter quand dans certains cas, une réification peut jouer le rôle de méta-objet, comme c'est le cas avec les classes dans les langages à classes.

Le deuxième aspect concerne les relations causales entre les niveaux, soit les interactions entre objets et objets du méta-niveau. Il faut notamment préciser le lien causal, qui précise l'arité de la relation objet / méta-objet.

Le point suivant s'intéresse aux règles de passages entre les niveaux, qui dépendent de la nature du MOP. Dans les MOP implicites, le changement de niveau est transparent pour l'utilisateur, les mécanismes d'exécution étant déclenchés à l'insu des objets du niveau de base. Ces derniers ne référencent pas explicitement le ou les méta-objets impliqués. Si le MOP est explicite, alors c'est au programmeur de définir les points de passage, en insérant dans un niveau les appels au niveau supérieur et inversement. Au niveau de base, le ou les méta-objets doivent donc être référencés et les association explicitées.

[4]Ainsi, l'acronyme MOP est redéfini en *Meta-level Object Protocol*, ou Protocole d'Objet du Méta-niveau

Enfin le dernier point concerne la gestion des régressions à l'infini. Si l'on considère que les objets du niveau de base sont contrôlés par ceux du méta-niveau, qu'en est-il pour ces derniers ? On peut ainsi remonter de niveau en niveau, à l'infini. Ce type de problème se rencontre notamment lors de la construction du système, avec par exemple la récursion du lien d'instanciation, sur lequel on reviendra ultérieurement. On parle alors de problème de *bootstrap*, ou amorçage, la solution de base étant qu'à un niveau donné l'objet assure son propre contrôle.

Ces aspects énoncés, il ne faut pas oublier que l'objectif d'un MOP est de faciliter la mise en oeuvre de langages adaptés aux spécificités d'un domaine, dans les limites de la raison d'être du MOP. Le tout doit se pouvoir se faire avec élégance et simplicité [81]. Autrement dit, si un MOP n'est pas une boîte noire puisqu'il admet des ajustements de la part de l'utilisateur, la solution d'une boîte blanche n'est pas adaptée non plus. Cela voudrait dire offrir au regard du programmeur toute la complexité de la mise en oeuvre, ce qui ne simplifierait en rien la création de langages. La solution réside dans le principe de la boîte grise, où l'utilisateur dispose d'une interface d'utilisation du système (les objets du niveau de base) et d'une interface d'adaptation du système [133]. Cette seconde interface doit alors énoncer clairement quels sont les mécanismes d'exécution que l'utilisateur peut ajuster et comment cela peut être réaliser, sans qu'il soit nécessaire de connaître la complexité du système sous-jacent.

4.3 Le langage de représentation des connaissances par objets archipel.lang

Archipel.lang dispose de deux protocoles simples permettant d'assurer l'ouverture nécessaire à la mise en oeuvre de langages de représentation adaptés. Le premier concerne l'interprétation des propriétés décrivant, dans un formalisme XML[5] [153], les concepts contextuels. Ce MOP offre une grande flexibilité dans la construction des objets et des relations entre objets. Le second est plus traditionnel, et s'attaque au contrôle de l'envoi des messages. Il permet ainsi de gérer avec finesse le comportement des objets constitués. Sa description nécessitera l'introduction d'un modèle objet Java nommé *CASO*, pour *Class As Object*, fortement inspiré du modèle ObjVLisp [33].

[5] *eXtensible Markup Language*

TABLEAU 4.1 – Comparaison des trois modèles de représentation de la connaissance dans archipel.lang

sens de la construction du langage ↑	représentation des items	utilité	disponibilité	évolution de la connaissance	visibilité	capacité d'ouverture
Modèle objet sémantique	modèle objet Java+CASO, interprétation sémantique du XML	utilisation de la connaissance à l'exécution (raisonnement, manipulation)	disponible à l'exécution uniquement	construction objet fixée	accessible aux autres composants du framework	MOP de contrôle de l'envoi des messages
Modèle objet structurel	modèle objet Java, de même structure que le XML	support pour la génération du modèle sémantique et pour l'édition graphique des connaissances	disponible à l'exécution uniquement	construction objet modifiable (simulation graphique de la sémantique)	interne à archipel.lang	MOP d'interprétation des propriétés XML
Modèle XML	schéma XML	support pour la persistance	représentation persistante	modifiable : sauvegarde connaissances éditées graphiquement ou modification manuelle	interne à archipel.lang	aucune restriction sur le contenu XML

4.3.1 Trois niveaux de représentation

La connaissance est présente sous trois formes au sein d'archipel.lang. Le *modèle XML* et le *modèle objet structurel* constituent les deux niveaux de représentation nécessaires à la mise en place du modèle objet final, dit *modèle objet sémantique*. C'est ce dernier qui sera utilisé par les autres composants du *framework* à des fins d'utilisation des connaissances contextuelles. Le tableau 4.1 propose une comparaison de ces trois modèles, détaillée dans cette section, et qui sera suivie de leur présentation technique.

Modèle XML La notion de base d'archipel.lang est l'*item*. Ce terme, volontairement générique, désigne toute connaissance représentée, quelle qu'en soit la nature. Dans le modèle XML, les items sont représentés selon un formalisme d'une grande simplicité : chaque item possède un identifiant, qui doit être unique dans une représentation donnée, éventuellement un nom ou une désignation, et un ensemble de propriétés. Les propriétés, qui décrivent la nature de l'item, sont présentées sous la forme de couples de chaîne de caractères non contraintes (nom, valeur). Le schéma XML correspondant, qui étend la représentation à une collection d'items, est donné à l'annexe A. Une représentation graphique partielle du schéma est visible à la figure 4.2.

Ce modèle constitue la couche de plus bas niveau de la mécanique. Il en est non seulement la représentation la plus persistante, mais aussi la plus ouverte. En effet, contrairement à un langage comme OWL, lui aussi basé sur une syntaxe XML et sur lequel on reviendra ultérieurement, les noms des balises du modèle XML ne témoignent en rien de la nature des items décrits. Il ne s'agit que d'un simple cadre structurel pour la description d'items, plus particulièrement à des fins de persistance. Ainsi, c'est l'interprétation

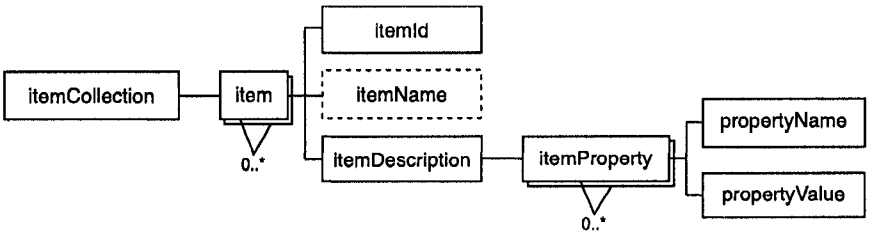

FIGURE 4.2 – Représentation graphique partielle du schéma XML pour la description des connaissances

des propriétés, de leur nom et de leur valeur, qui permettra de donner au modèle objet final la sémantique de la connaissance décrite en XML. En somme, ce schéma ne fait aucune présupposition sur ce que sera la représentation par objets : il est universel.

Modèle objet structurel Pour constituer le modèle objet sémantique à partir du contenu XML, le passage obligé est le modèle objet structurel. Ce modèle est dit structurel, car l'organisation de ses classes est similaire à l'organisation des balises dans le schéma XML. Néanmoins, ce modèle ne se limite pas à être une version objet du contenu XML. En effet, il associe aux connaissances décrites divers objets qui vont servir de support pour la création du modèle sémantique, notamment pour l'écriture des sources Java et le chargement des objets suite à la compilation dynamique des sources.

Toute cette mécanique repose sur l'interprétation des propriétés décrites en XML, réalisée grâce au premier MOP d'archipel.lang. Cette interprétation confère de plus aux objets du modèle structurel la capacité de représenter graphiquement la connaissance. Ainsi, archipel.lang dispose d'outils graphiques permettant à l'utilisateur de travailler sur les connaissances contextuelles sans passer par le XML et en disposant d'une vue quasiment sémantique de la connaissance décrite.

Modèle objet sémantique Le modèle sémantique est la concrétisation de cette démarche. Il présente une modélisation objet dont la structure est conforme à la sémantique décrite en XML. Basé sur Java, le modèle CASO et le MOP de contrôle de l'envoi de message, il offre aux composantes du *framework* Archipel une solution ouverte et intégrée pour utiliser la connaissance. Néanmoins, ce modèle a une profonde limitation : il est parfaitement figé. C'est pourquoi l'édition des connaissances contextuelles, par exemple lors de la configuration de l'orthèse, doit passer par le modèle objet structurel. Ce dernier offre la

souplesse nécessaire pour réaliser efficacement la description d'un contexte, ce qui n'est forcément chose aisée.

4.3.2 Le MOP d'interprétation du contenu XML

L'objectif de ce premier MOP est d'offrir à l'utilisateur la possibilité de construire une représentation objet adaptée à son domaine, à partir d'une spécification XML de la connaissance et par l'intermédiaire d'une représentation objet structurelle de celle-ci. Ce protocole est particulier dans la mesure où une fois le modèle objet sémantique créé, le modèle structurel devient caduc. En somme les objets qui constituent ce modèle ne seront jamais manipulés dans un contexte programmatique autre que pour la mise en place du modèle sémantique, ainsi que pour l'édition graphique de la connaissance.

De XML au modèle objet structurel

La figure 4.3 donne la représentation graphique d'un document XML construit à partir du schéma introduit précédemment. Le document XML correspondant est donné à l'annexe B. Sont présentées quelques connaissances issues du contexte matériel du scénario. On y retrouve le concept de "contenant à boisson", de "tasse", ainsi qu'une "tasse bleue" et une "tasse rouge".

L'item d'identifiant aat :300194567[6], c'est-à-dire le "contenant à boisson", a pour description une propriété sans valeur dont le nom est "itemClass". Par là, on sous-entend que l'entité représentée est un concept abstrait, sensé prendre la forme d'une classe dans un modèle objet. L'item "tasse" est dans la même situation, une relation supplémentaire la concernant étant décrite, avec la propriété "extends" dont la valeur est l'identifiant du "contenant à boisson". Autrement dit, une relation d'héritage est posée entre le contenant à boisson et la tasse. C'est tout du moins la sémantique que l'utilisateur associe à la propriété "extends", le choix du nom de la propriété étant d'ailleurs purement subjectif. S'agissant des items "tasse bleue" et "tasse rouge", la propriété "instanceof" dont la valeur fait référence à l'identifiant du concept de tasse dénote le caractère concret de ces objets. Il va s'agir d'instances, qui vont en l'occurrence se différencier par leur décoration. Cette caractéristique est d'ailleurs issue de la définition de leur classe, comme traditionnellement dans un modèle classe-instance. La question est donc la suivante : comment interpréter ces

[6]La logique d'identification des items est présentée à l'annexe C

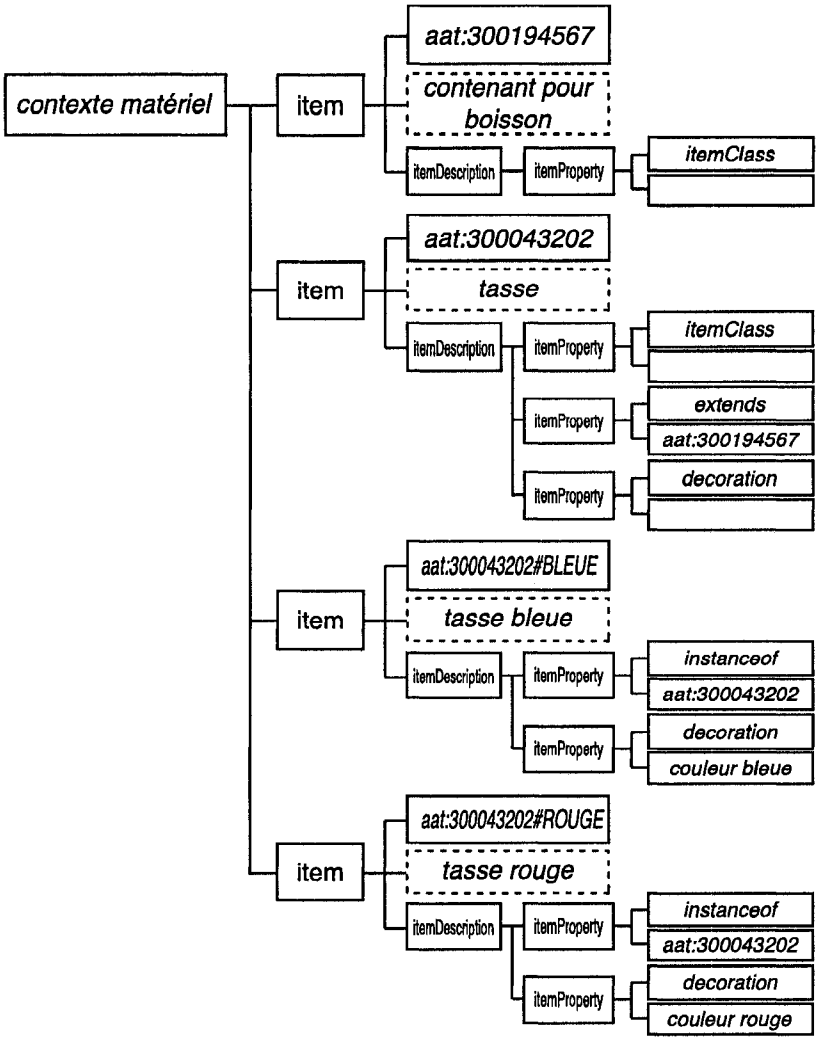

FIGURE 4.3 – Représentation graphique d'un document XML de description de connaissances contextuelles. Les termes en italiques sont les valeurs données aux éléments XML correspondant dans le schéma 4.2

propriétés, dont un échantillon très limité est donné ici, pour construire dynamiquement des objets sémantiquement conformes à ce que l'utilisateur a décrit en XML ?

Interprétation des propriétés

Le processus d'interprétation des propriétés est confié à un ensemble de stratégies. Le rôle d'une stratégie d'interprétation est, pour une propriété donnée, de garantir la génération d'un modèle objet reflétant la sémantique XML. L'association stratégie/propriété se fait sur la base du nom de la propriété, la valeur étant logiquement prise en compte lors de l'interprétation. Par exemple, l'application de la stratégie associée à la propriété "itemClass" du concept "tasse", qui en l'occurrence n'a pas de valeur, doit avoir pour effet la mise en oeuvre d'une classe Java modélisant ce concept. Cela sous-entend une écriture du code adaptée.

La sémantique des stratégies rappelle celle des *facettes* dans les langages de *frames* (ou *schémas*). Introduits en 1975 par M. Minsky, ces langages représentent une étape importante dans l'histoire de la représentation des connaissances. Un *frame* est "une entité générique composée d'attributs qui décrivent les différentes propriétés du concept représenté. Un attribut est à son tour décrit par un certain nombre de facettes, possédant des valeurs. Les facettes expriment des modalités descriptives ou comportementales, représentant différents points de vue sur l'attribut. **Elles servent non seulement à décrire la nature de l'information que l'attribut contient, mais aussi à préciser comment la calculer ou l'utiliser**" [100]. Bien que la comparaison s'arrête là puisque les facettes sont avant tout dédiées à l'inférence des valeurs des attributs, cette définition, prise hors contexte, s'applique plutôt bien aux stratégies d'interprétation des propriétés.

La prochaine étape est donc la mise en place d'une infrastructure de travail pour les stratégies. Celle-ci va être trouvée dans le modèle objet structurel. Comme présenté par le diagramme de classes 4.4, ce modèle n'est qu'une transposition objet du schéma XML. Il va donc permettre de recueillir sous forme d'objets Java les données du document XML. L'analyse syntaxique est alors assurée par le *framework* applicatif domus.xml, présenté à l'annexe D. Néanmoins, le modèle structurel apporte une plus-value : il associe un *état* à chaque objet Item et ItemProperty, cette dernière classe représentant les propriétés (nom et valeur). L'état d'un Item est modélisé par un objet de type ItemState, celui d'un ItemProperty par un objet de type ItemPropertyState. Ce sont les états qui vont servir de support pour la génération du modèle objet sémantique, et leur détail sera donné ultérieurement. En guise d'exemple, c'est l'état associé à l'instance d'ItemProperty représentant la pro-

priété "itemClass" dans le modèle structurel qui devra garantir l'écriture d'un code source Java de type classe. Ce que l'on retiendra surtout ici, c'est que l'affectation des objets états est confiée aux stratégies d'interprétation des propriétés.

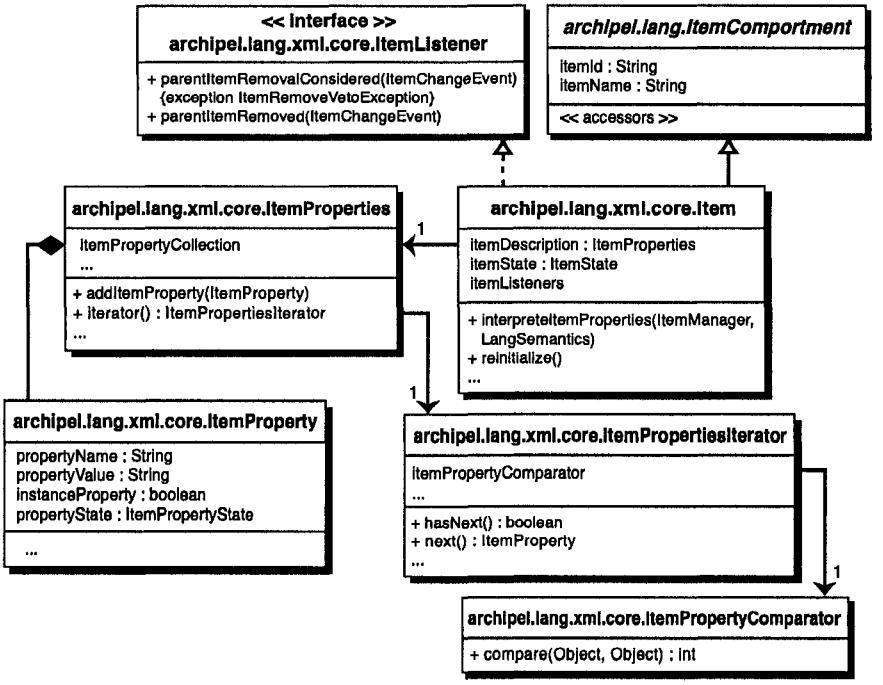

FIGURE 4.4 – Diagramme de classes, modèle objet structurel

Principe d'une stratégie d'interprétation

Les stratégies d'interprétation répondent au *patron de conception* "stratégie". Rappelons que les patrons de conception, plus connus sous leur appellation anglo-saxonne *design pattern*, sont une "capitalisation d'expériences ayant comme but de codifier des solutions considérées comme "bonnes" à certains problèmes récurrents de conception" [110]. Les plus classiques sont les patrons "GoF", issus du travail de messieurs Gamma, Helm, Johnson et Vlissides[7] [57], et dont fait partie le patron de conception "stratégie". En l'occurrence, le problème est de pouvoir constituer une famille d'algorithmes répondant à une même problématique - ici l'interprétation des propriétés -, et de pouvoir changer d'algorithme - en fonction de la propriété - sans modifier le code hôte - la boucle

[7] "GoF" pour *Group of Four* ...

générale d'interprétation des propriétés des items d'une collection -. La solution réside dans le découplage de l'algorithme et du code hôte, ce dernier ne travaillant qu'avec une abstraction du premier (une interface Java en l'occurence).

La notion d'état d'un objet Item ou ItemProperty fait référence au patron de conception GoF éponyme. L'idée sous-jacente est que le comportement d'un objet peut dépendre en tout ou en partie de son état, ce dernier - et donc *a fortiori* le comportement - pouvant changer à l'exécution. C'est le cas de l'exemple donné sur la génération du code source pour le modèle objet sémantique : l'écriture du code va dépendre de l'état d'un objet Item et des ItemProperty qui lui sont associés. Pour répondre efficacement à cette problématique, les comportements dépendant de l'état sont isolés de l'objet parent sous la forme d'une interface abstraite de programmation, qu'implémenteront un ensemble d'objets représentant les différents états possibles.

Originellement, l'état de tout objet Item et ItemProperty est dit "inconnu". Trois types de transitions peuvent alors intervenir, comme l'illustre la figure 4.5. La première correspond à l'application d'une "méta-stratégie". Une telle stratégie devrait être associée à toute propriété dont le rôle est d'indiquer la place et le comportement de l'item dans le modèle objet sémantique. Le nouvel état fera de cette propriété une "méta-propriété". C'est le cas de la propriété "itemClass", qui fait de l'item une future classe Java. Cette transition peut alors avoir un effet sur l'état de l'item et l'état des autres propriétés. En l'occurrence, l'item va prendre l'état de classe. Pour ses autres propriétés, cela va dépendre si une stratégie particulière leur est associée. Si tel n'est pas le cas, leur état passera à celui d'attribut de type String[8]. Le second type de transition correspond aux propriétés servant de support pour la connaissance et devant être "transformées" en attribut Java, mais avec un type de données spécifique. Celui-ci est en quelque sorte précisé par l'état. Enfin, le troisième type de transition correspond à l'échec de l'interprétation, quelque soit la nature de la stratégie, faisant passer l'objet modélisant la propriété dans un état d'erreur. Cela devrait avoir pour conséquence de faire passer aussi l'objet Item dans un état d'erreur. Si les propriétés sont modifiées par la suite, alors une nouvelle interprétation pourra confirmer l'état courant ou le modifier. A noter qu'il sera précisé ultérieurement comment une stratégie est associée à une propriété.

L'utilisation de patrons de conception assure l'ouverture du langage initiée avec le

[8]Le type Java String, i.e. chaîne de caractères, est considéré comme le type par défaut d'un attribut d'une classe dans le modèle objet sémantique, dans la mesure où les valeurs des propriétés dans les documents XML sont de ce type.

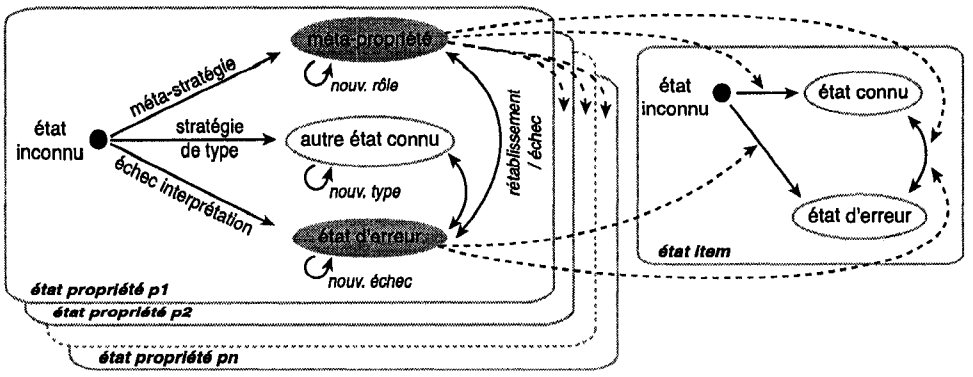

FIGURE 4.5 – Diagramme état-transition, état des objets représentant un item et ses propriétés dans le modèle structurel. Les flèches en pointillées représentent l'incidence possible d'une transition vers les états sur fond gris

modèle XML. On offre ainsi la possibilité à l'utilisateur de créer des stratégies d'interprétation et des états en fonction des besoins de la représentation. Reste la contrainte d'un modèle à objet, mais celui-ci offre un grand nombre de variantes, tant au niveau représentation que manipulation : langage à classes, à métaclasses, à métaclasses "complet", à prototype, introduction de patrons de conception. Ces approches, qui seront introduites ultérieurement, offrent toutes des caractéristiques différentes en terme de représentation ou de manipulation des connaissances. En somme, ce n'est pas "un point unique dans l'espace global des conceptions de langages" de représentation et de manipulation par objets, mais plutôt "une région entière à l'intérieur de cet espace" qui va pouvoir être créée [82]. C'est l'esprit des protocoles à méta-objets.

Description du MOP d'interprétation

Le MOP d'interprétation est construit autour des objets ItemState et ItemPropertyState, modélisant l'état associé aux objets Item et ItemProperty dans le modèle structurel. Comme l'indique le diagramme de classes 4.6, la spécification de ces objets est réalisée sous la forme d'interfaces Java. On rappellera qu'en tant que spécification abstraite, une interface ne donne que la signature des méthodes que les classes concrètes modélisant les différents état possibles doivent mettre en oeuvre. En l'occurrence, ces signatures ont pour objet l'association à un état des différents concepts représentant les objets du méta-niveau de ce MOP.

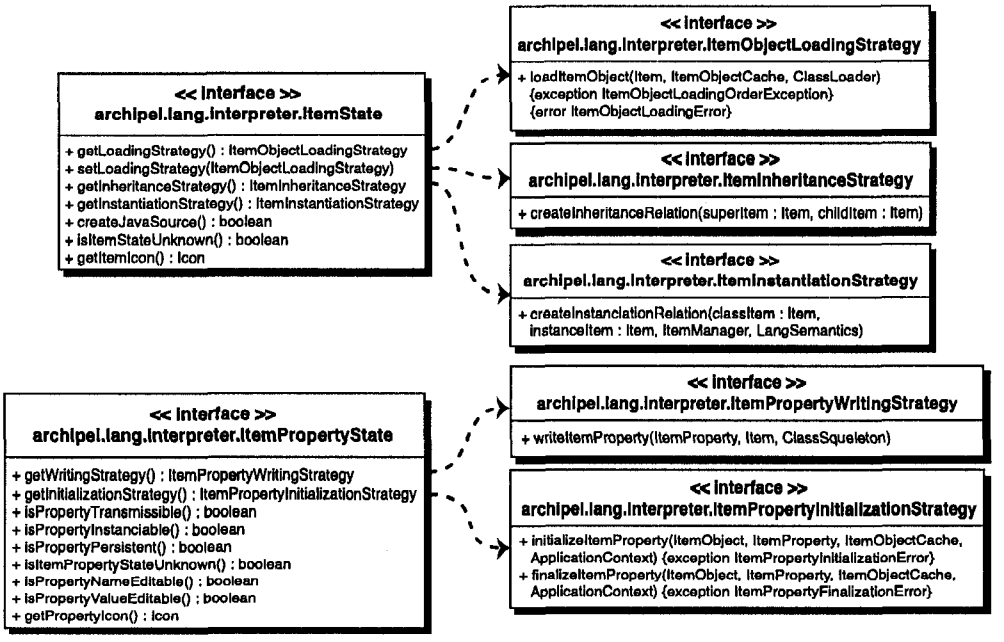

FIGURE 4.6 – Etat des items et propriétés et stratégies associées : diagramme de classes

Le tableau 4.2 décrit les différentes facettes du MOP. Celui-ci est décomposé en quatre thématiques qui s'inscrivent dans un objectif de création de la connaissance ou de création du modèle objet sémantique. Une lecture de ce tableau est proposée ici.

Thématique : établir des relations entre items Sans l'établissement de relations entre items, la connaissance décrite perd la plus grande part de sa richesse. De par l'approche objet choisie, il deviendrait en effet difficile de raisonner sur ces concepts. L'héritage permet de voir une tasse comme un contenant à boisson, au même titre qu'un bol lors du suivi de la réalisation d'une activité. La relation d'instanciation offre au système la possibilité de percevoir l'utilisation de la "tasse bleue" sous le même angle sémantique que celle de la "tasse rouge".

En Java, un troisième type de relation peut être établi : l'implémentation. L'implémentation est le lien qui unit une classe à une interface Java. On a rappelé précédemment qu'une interface Java était une spécification comportementale abstraite, soit un ensemble de signatures de méthodes qu'une classe doit mettre en oeuvre. Concrètement, cela permet de voir les instances de la classe selon le point de vue défini par l'interface. L'horizon relationnel est ainsi grandement ouvert, Java étant un langage à héritage simple : une

TABLEAU 4.2 – Description du MOP d'interprétation du contenu XML

thématique	objectif	objets du méta-niveau : réifications	objets du méta-niveau : méta-objets	technique d'adaptation	support offert par le framework	relation entre niveau	objet du niveau de base
établir des relations entre items (héritage, instanciation, implémentation)	création de la connaissance	– propriété héritable ? – propriété instanciable ? (valeurs booléennes)		modification valeur		explicitement précisée par l'état de la propriété	propriété d'un item
			– stratégie de création d'une relation d'héritage – stratégie de création d'une relation d'instanciation ou d'implémentation	patron stratégie	stratégies usuelles pour un modèle objet à métaclasses	explicitement précisée par l'état de l'item	item
créer les sources Java	création du modèle objet sémantique	squelette de classe	stratégie de création (écriture) de la source Java	patron stratégie	stratégies de création de classes et d'interfaces respectant encapsulation et patron de nommage	explicitement précisée par l'état de la propriété	propriété d'un item
		représente un source Java ? (valeur booléenne)		modification valeur		explicitement précisée par l'état de l'item	item
gérer le cycle de vie (chargement, initialisation, préparation à la persistance)	création du modèle objet sémantique	propriété persistante ? (valeur booléenne)	stratégie d'initialisation et de finalisation (préparation à la persistance)	modification valeur & patron stratégie	stratégie générique pour tous types de propriété *n'utilisent que l'envoi de messages*	explicitement précisée par l'état de la propriété	propriété d'un item
		propres au langage Java	stratégie de chargement	patron stratégie	stratégies pour chargement (méta)classes et instances	explicitement précisée par l'état de l'item	item
manipulation graphique de la connaissance	création de la connaissance		agencement graphique pour la représentation de relations relation pouvant être créées à partir de l'item	patron stratégie patron visiteur basé sur les stratégies de création de relation	stratégie d'agencement hiérarchique stratégies usuelles pour un modèle objet à métaclasses	explicitement précisées par les stratégies d'interprétation des propriétés de l'item	item
			éditeur de propriété	héritage	éditeurs pour les types usuels de propriété	explicitement précisée par la stratégie d'interprétation de la propriété	propriété d'un item

classe ne peut hériter que d'une seule autre classe.

Sous l'angle de la représentation des connaissances, la relation d'implémentation, qui sémantiquement ne témoigne pas d'une relation d'ordre comme c'est le cas avec l'héritage, va permettre d'exprimer des relations qui n'entrent pas sous la coupole de ce dernier. C'est le cas avec la notion de service d'interaction homme-machine s'appliquant aux effecteurs. Le concept de service d'IHM va pouvoir, d'un point de vue Java, être représenté sous la forme d'une interface. Il va d'ailleurs certainement s'agir d'une interface de marquage, soit une interface vide, sans signature de méthodes à mettre en oeuvre, dont l'utilité se limite à marquer de sa sémantique les classes qui l'implémentent.

Cette remarque sur Java est importante, dans la mesure où l'on veut assurer la compatibilité du langage de RC avec le langage de programmation du *framework*. Néanmoins, il a été dit en début de chapitre que le modèle objet de Java limitait son expressivité. Cette limite se fait par exemple sentir lorsqu'il question de décrire les concepts abstraits. Dans le scénario, Pierre utilise un localisateur d'objet, c'est-à-dire une interface graphique composée de vignettes représentant divers concepts ou objets matériels présents dans son environnement. La sélection d'une vignette - l'interface est présentée sur un écran tactile - entraîne un acte de mise en évidence de l'élément choisi dans l'environnement. La question est alors de savoir comment faire le lien entre une vignette et l'élément à rechercher. Logiquement, la vignette ne devrait être qu'une représentation visuelle de l'objet ou du concept en question, cette représentation devant faire partie de sa description, avec pour valeur un nom de fichier de type image, par exemple. Si parmi ces concepts et objets se trouvent le concept de tasse ou même de contenant à boisson, l'idée étant très certainement de localiser le placard dans lequel sont généralement rangés tous les objets de cette nature, alors le nom du fichier image devra être trouvé auprès du concept, et non auprès de l'une de ses instances. Cette illustration s'appliquera ainsi à tous les objets de ce type. Pour cela, selon le modèle classe-instance, l'attribut "vignette" devra être défini au niveau de la classe du concept, qui est lui-même une classe. On parle alors de métaclasse, soit la classe d'une classe. Or cette capacité de représentation, bien que présente en Java, est verrouillée, répondant ainsi à un besoin de sécurité du langage sur lequel on reviendra. C'est pourquoi une sur-couche Java s'impose pour répondre aux besoins de représentation des connaissances. Elle sera fournie par le modèle CASO, qui sera présenté ultérieurement.

A ce stade de l'analyse, on fera donc abstraction des considérations liées à la mise en oeuvre du langage de RC en Java. Ainsi, la première thématique de ce MOP d'in-

terprétation, qui s'intéresse à l'établissement des relations entre items, pourra être considérée au-delà des possibilités de Java. On restera toutefois dans les limites d'un modèle objet complet, ce qui sera précisé par la suite. Ce qu'il faut retenir ici, c'est que l'établissement de relations se concrétise par un échange de propriétés entre items. Par exemple, la relation d'instanciation liant la "tasse bleue" au concept de "tasse" va avoir pour conséquence l'introduction de la propriété "décoration" dans la description de l'item "tasse bleue". Cette manipulation découle de la sémantique de la relation, elle n'a pas à être réalisée à la main. Pour ce faire, le contrôle de cet échange est assuré par les différentes stratégies constituant les méta-objets du MOP. Ces stratégies sont associées à l'état de l'Item. Par exemple, l'item "tasse" qui représente une classe va disposer d'une stratégie permettant la transmission de ses propriétés lorsqu'un lien d'instanciation est établi. A noter que toutes les propriétés ne doivent pas être transmise : "itemClass" est une méta-propriété, elle ne décrit en rien les attributs de la classe "tasse". Pour cela, les réifications portées par l'état des propriétés sont utilisées. Il ne s'agit que de valeurs booléennes (la propriété doit-elle être prise en compte dans une relation d'instanciation ?), mais cela est parfaitement suffisant.

Pour terminer, on notera que les différentes stratégies proposées sont, comme leur nom l'indique, une mise en oeuvre du patron de conception éponyme. C'est la garantie d'un contrôle ouvert sur la structure et le comportement des objets du niveau de base que sont les Item et les ItemProperty, dans l'optique de création du modèle objet sémantique.

Thématique : création des sources Java La création du modèle sémantique passe par l'écriture dynamique du code Java adéquat, qui sera par la suite compilé. La principale réification utilisée est un objet représentant un squelette de classe. Ses différentes composantes (imports, en-tête, déclarations des attributs, déclarations des accesseurs) sont renseignées par l'intermédiaire de stratégies d'écriture associées aux états des ItemProperty. Les méta-propriétés auront principalement une influence sur l'en-tête de la classe. Les autres préciseront généralement le type de l'attribut, l'import pour ce type si nécessaire. Le tout se fait dans le respect des conventions de nommage du langage hôte et du principe d'encapsulation des attributs.

Cette mécanique n'a pas vocation à permettre l'écriture d'un code source comprenant des méthodes destinées à effectuer des traitements, autres que ceux liés à l'accès en écriture et en lecture aux attributs. Cela correspond à l'objectif premier de représentation de connaissances. Néanmoins, le comportement des objets sémantiques créés est plus

subtil, dans la mesure où intervient à ce niveau le second MOP, lié au contrôle de l'envoi des messages. Celui-ci sera introduit par la suite. Cet aspect mis à part, on pourrait parfaitement envisager une propriété XML dont la valeur serait le corps d'une méthode, autrement-dit du code Java. Une stratégie d'écriture adaptée devrait alors être mise en oeuvre.

Thématique : gestion du cycle de vie Une fois les sources Java compilées, celles-ci doivent être chargées pour être accessibles au sein de l'espace d'exécution géré par la machine virtuelle Java. Une fois chargées, les classes peuvent être instanciées puis leurs instances initialisées. L'ensemble du modèle objet sémantique est alors créé.

La compilation est effectuée grâce aux outils fournis par l'environnement de développement Java. Le chargement, terme encapsulant le chargement des classes mais aussi l'instanciation des objets concrets, est réalisé selon une stratégie associée à l'état de l'objet Item. S'agissant des classes, les outils usuels Java (ClassLoader) sont utilisés. Du côté des instances, tout passe par l'envoi de message, protocole de base du modèle CASO pour l'interaction entre objets. Il en est de même pour l'initialisation qui repose sur des stratégies associées aux états des ItemProperty. Cette mécanique sera donc présentée ultérieurement.

Enfin, on notera que le cycle de vie comprend la gestion de la persistance des items. Celle-ci est réalisée par des stratégies dites de finalisation, et utilise à nouveau le modèle objet structurel. C'est en effet lui qui sert d'intermédiaire pour le modèle XML. La gestion de la persistance correspond donc essentiellement à l'affection des nouvelles valeurs prises par les attributs des objets, le modèle sémantique n'ayant pas vocation à créer de la connaissance mais plutôt à permettre son utilisation à des fins de raisonnement et de manipulation.

Thématique : manipulation graphique de la connaissance Même si les documents XML décrivant les connaissances restent parfaitement éditables, la manipulation de la connaissance sous ce format est délicate. C'est pourquoi un outil graphique d'édition, fonctionnant sur la base du modèle objet structurel, a été développé.

La figure 4.7 présente une copie d'écran de l'éditeur, présentant les connaissances introduites à la figure 4.3. Sur la partie gauche de l'interface, les items "contenant à boisson", "tasse" et les deux instances sont présentés selon une organisation hiérarchique. Il s'agit d'une représentation traditionnelle pour un langage à classes. Néanmoins, cette

construction graphique n'est qu'une simulation visuelle de la sémantique, puisque les objets présents en arrière-plan sont tous de même nature, instances de la classe Item. Le contrôle est réalisé pour chaque item par une stratégie d'agencement, généralement associée aux méta-propriétés. L'outil graphique permet de plus l'ajout de nouvelles connaissances, et ce selon la sémantique des concepts représentés. Un patron de conception visiteur basé sur les stratégies de création de relations est alors utilisé. Enfin, les ItemProperty sont éditées selon un principe de feuille de propriétés. Une explication plus complète de toute cette mécanique d'édition et des concepts sous-jacents peut être trouvée à l'annexe F.

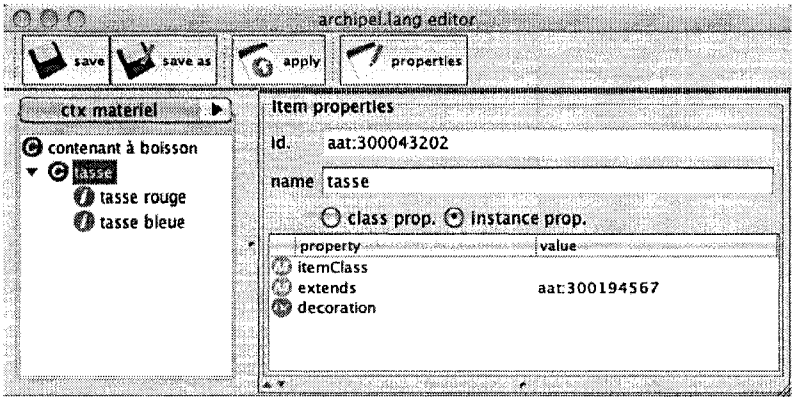

FIGURE 4.7 – Edition graphique des connaissances

On terminera cette présentation du MOP d'interprétation en précisant que le *framework* archipel.lang fournit un grand nombre de mises en oeuvre des stratégies impliquées dans le MOP, ainsi que de nombreux outils complémentaires, notamment pour l'écriture dynamique des sources.

Association des stratégies d'interprétation aux propriétés

Avant de présenter le modèle CASO et le MOP de contrôle de l'envoi des messages, il reste à préciser comment les stratégies d'interprétation, qui définissent l'état des objets Item et ItemProperty au sein du modèle structurel, portant eux-mêmes la plupart des méta-objets du protocole, sont associées aux propriétés telles que définies en XML.

L'explication est relativement simple : archipel.lang est auto-décrit. Autrement dit, la sémantique du langage défini par l'utilisateur, qui dépend des stratégies d'interprétation associées aux propriétés des items, est décrite comme toute autre connaissance, dans le

même formalisme et selon la même mécanique. Dès lors, cette connaissance doit elle-même être interprétée. Le problème de régression en résultant est résolu très simplement : la sémantique du langage permettant de spécifier celle du langage utilisateur est fixée en dure et ne peut être modifiée. Elle représente l'amorçage de la mécanique d'interprétation. Une explication plus complète de cette mécanique, mettant bout-à-bout les éléments présentés jusqu'à présent, sera donnée à la fin de ce chapitre, car elle nécessite l'introduction du modèle CASO et du second MOP d'archipel.lang.

4.3.3 Le MOP de contrôle d'envoi de messages

Comme cela a déjà été dit, Java ne permet pas la création de métaclasses, c'est-à-dire d'objets ayant la capacité de créer des instances étant elles-mêmes des classes. Or la représentation des connaissances contextuelles nécessite un tel niveau de modélisation. Cette approche est de plus très utile à des fins de raisonnement, notamment lorsqu'il est question d'avoir connaissance des instances d'une (méta)classe. Il faut donc offrir au langage hôte la sur-couche nécessaire. L'inspiration sera trouvée dans le modèle ObjVLisp, qui est présenté dans la prochaine section. L'ensemble permettra la mise en place d'un MOP "traditionnel" pour le contrôle d'envoi de messages.

L'élégance du modèle ObjVLisp pour les langages à métaclasses

La programmation par objets et les langages sous-jacents étaient, dans les années 80, en plein essor mais aussi en pleine gestation. Parmi les concepts nouveaux qui ont alors fait leur apparition, celui de métaclasse traduisait la volonté d'uniformité dans les langages à classes. C'est la fameuse *querelle des Universaux*, qui opposa Aristote à Platon, et qui se pose en des termes renouvelés dans l'approche objets : "il s'agit de savoir si les concepts existent réellement dans le monde (position dite réaliste[9]) ou s'ils n'apparaissent que dans notre discours sur le monde (position nominaliste), de savoir s'ils sont des choses ou seulement des mots. Dans notre langage informatique, cela revient à savoir si les classes sont des objets (manipulables comme tels) ou seulement des textes (plus ou moins) compilés" [119].

La position réaliste n'est pas à discuter ici : il est nécessaire de "chosifier" les concepts abstraits pour répondre aux besoins de représentation. Or c'est le lien d'instanciation qui confère aux instances la qualité d'objets : ils sont instances d'une classe. Pour of-

[9]Celle de Platon, qui donne aux concepts abstraits la qualité de sujet.

frir ce même statut aux classes, il leur suffit d'être instance d'une autre classe, baptisée *métaclasse*. Et cette réification rentre parfaitement dans l'approche réflexive qui a été - partiellement - introduite précédemment, avec notamment son lot de problème de régression.

Dans un langage éminemment objet comme Smalltalk, la notion de métaclasse facilite la définition du comportement d'une classe, car celle-ci est alors vue comme un objet "standard". Par "standard", on entend la capacité d'un objet à recevoir et à interpréter des messages - vu comme l'unique moyen d'interaction entre objets. L'interprétation faite du message - pouvant aller jusqu'à sa non-compréhension - traduit alors le comportement de l'objet. Néanmoins, la relation de "méta" instanciation n'est pas libre : c'est le système qui, à chaque classe créée, lui associe automatiquement une métaclasse, imposant ainsi une relation 1 à 1 entre les niveaux [33].

En Java, la notion de métaclasse est elle aussi présente, mais la mécanique est complétement verrouillée. Il n'existe en effet qu'une seule métaclasse, la classe **java.lang.Class**, dont chaque classe est implicitement instance. Aucune autre métaclasse ne peut être créée par l'utilisateur. La raison en est simple : en offrant à toute classe une légitimité dans le système, on lui offre aussi le pouvoir de définir son comportement ou redéfinir celui dont elle hérite, selon une logique qui va être précisée dans cette section. Si ce comportement concerne la création d'objet, alors tout devient possible, y compris l'introduction de failles de sécurité. De par les domaines applicatifs de Java, il est plus sage d'empêcher cela.

Néanmoins, pour le présent problème de représentation de connaissances, il est intéressant de pouvoir définir des métaclasses en toute généralité, afin de pouvoir organiser sans aucune contrainte l'information. ObjVLisp, modèle issu des travaux de P. Cointe et J.-P. Briot [33], proposé en 1987, affronte ce problème sous l'angle suivant : quelle doit être l'architecture minimale d'un langage à métaclasses pour que l'ensemble des concepts (instances terminales, classes et métaclasses - admises comme la couche d'abstraction des classes -) soit uniformément manipulable ? La réponse tient en six postulats :

1. Un *objet* est un ensemble de connaissances, représentées par ses données, et d'aptitudes, décrites par ses méthodes[10] ;
2. L'unique protocole pour actionner un objet est l'envoi de *message*, un message spécifiant le nom de la méthode à invoquer et les arguments à passer ;

[10]L'auteur utilise le terme *procédure*, celui-ci décrivant en fait la nature d'une méthode. Aujourd'hui seul ce second terme reste [119].

❸ Chaque objet appartient à une *classe*, qui spécifie ses données - ses attributs - et son comportement, défini par l'ensemble de ses méthodes. La classe est un modèle à partir duquel sont générés dynamiquement les objets, qui sont alors nommés *instances* de la classe. Toutes les instances d'une classe partagent la même structure, définie par ce modèle, mais divergent par la valeur associée à leur attributs, qualifiés de *variables d'instance* ;

❹ Une classe est elle aussi un objet, instanciée par une autre classe appelée sa *métaclasse*. Par conséquent (❸), à chaque classe est associée une métaclasse qui décrit son comportement en tant qu'objet. Il existe une métaclasse primitive, appelée Class, qui est instance d'elle-même ;

❺ Une classe peut être définie comme une sous-classe d'une ou plusieurs autres classes. Ce mécanisme, appelé *héritage*, permet le partage entre ces classes des variables d'instances et des méthodes. La classe Object représente le comportement le plus commun, partagé par tous les objets ;

❻ En complément des variables d'instances qui définissent l'environnement local d'un objet, existent les *variables de classe*, définissant un environnement global partagé par l'ensemble des instances d'une même classe. Ces variables de classe sont définies au niveau de la métaclasse, selon la logique suivante : les variables d'instance d'une classe sont les variables de classe de ses instances.

La mise en oeuvre de ces postulats se résume, dans le dialecte Lisp choisi, à deux petites pages de codes. On y retrouve les classes Class et Object qui, en tant que racines respectives des arbres d'instanciation et d'héritage, font partie de l'amorçage du système. La structure de tout objet est clairement définie, celle-ci permettant notamment de prendre trace des relations d'instanciation et d'héritage, indispensables à l'interprétation des messages. Le "new", message unique de création de tout objet du système, joue comme il se doit un rôle de premier plan, déléguant la partie initialisation de l'objet à une méthode *ad hoc* supportée par Object. Redéfinie par Class, qui hérite naturellement d'Object (❺), cette méthode permet de différencier à l'instanciation les classes des instances terminales. De plus, il est possible de définir des métaclasses par la même mécanique, le fait qu'une classe hérite de Class et non d'Object déterminant la méta-sémantique. Au final, le modèle est uniforme, complet, sa relative simplicité permettant de conclure quant à son élégance.

Le modèle CASO

Il a déjà été dit que Java, langage réaliste, n'en était pas moins incomplet, puisqu'il y est impossible de définir des métaclasses en toute liberté[11]. L'objectif d'archipel.lang est d'offrir une mécanique simple de langage à métaclasses complet, "à la" ObjVLisp, basé sur le langage hôte Java, sans pour autant modifier ce dernier. L'introduction d'un niveau méta explicite en Java a déjà été abordée dans la littérature, mais dans des optiques totalement différentes et par le biais de transformations de la plateforme Java. MetaXa, projet aujourd'hui abandonné, permettait la définition de méta-objets contrôlant différentes opérations de la machine virtuelle Java (gestion des messages, accès aux variables d'istances, (dé)verrouillage des objets, chargement des classes, création d'objets) [59]. La réflection était basée sur le système et non le langage. La prise de contrôle avait ainsi été rendu possible par une modification de la machine virtuelle. OJ (anciennement OpenJava) permet l'utilisation de macros en Java. Des méta-objets sont utilisés pour représenter la structure des objets, afin de gérer les transformations résultant de la mise en oeuvre d'une macro [146]. OJ nécessite une augmentation de la grammaire Java et l'utilisation d'un préprocesseur.

En Java, considérer une classe comme un modèle à objets n'est bien entendu pas un problème. La considérer comme un objet instance d'une métaclasse définie par l'utilisateur l'est. L'idée est de s'attaquer à ce second problème tout en conservant ses capacités premières, autrement dit de voir la classe comme un objet, instance d'une classe Java, et de lui ajouter la capacité de créer des instances. La solution est très simple et est présentée au sein du modèle CASO (pour Class AS Object). Voici l'interprétation des postulats d'ObjVLisp pour ce modèle.

❶ Tout élément du modèle CASO est un objet défini par ses données et méthodes, que l'on appelle *itemObjet*. Du point de vue du langage hôte, un itemObjet est l'instance terminale d'une classe Java.

❷ Le protocole de base pour actionner un itemObjet est l'envoi de message. Un message spécifie le comportement à invoquer, qui peut ou non correspondre au nom d'une méthode Java de l'itemObjet, et les arguments à passer. Si elles sont connues, il reste possible d'invoquer directement les méthodes Java d'un itemObjet. Cependant, il est admis que l'invocation directe d'une méthode Java et l'envoi d'un message de même nom puissent ne pas produire le même comportement. Ainsi, il

[11] En supposant que cela représente un besoin pour l'utilisateur ...

est postulé que seul l'envoi de message assure l'uniformité du comportement des itemObjets.

❸ Chaque itemObjet appartient à un CASO, dont il est l'instance. Toutes les instances d'un CASO partagent la même structure mais divergent par la valeur de leurs variables d'instances. Cette structure est définie par la classe réelle du CASO, dont le type est java.lang.Class.

❹ Un CASO est lui aussi un itemObjet, instancié par un autre CASO appelé métaCASO. Il existe un métaCASO primitif, ItemClassCASO, dont la classe réelle est archipel.lang.ItemClass. ItemClassCASO est instance de sa classe réelle.

❺ Un CASO peut être défini comme sous-CASO d'un autre CASO. Cette mécanique d'héritage permet le partage entre CASO des variables d'instances et méthodes et est mise en oeuvre entre les classes réelles des CASO conformément aux caractéristiques de l'héritage en Java. Si la classe réelle d'un CASO est une interface Java, alors celui-ci peut être implémenté par d'autres CASO. Il y a alors partage des méthodes et la mise en oeuvre entre classes réelles est conforme à l'implémentation des Interfaces en Java. Le CASO ItemObjectCASO représente le comportement le plus commun, partagé par tous les itemObjets. Sa classe réelle est archipel.lang.ItemObject.

❻ Même principe que pour ObjVLisp, les variables *static* du langage hôte, sensées offrir une notion de variable de classe, n'étant pas considérées dans le modèle CASO.

La classe archipel.lang.ItemObject ItemObject est la classe réelle de la racine de l'arbre d'héritage, ItemObjectCASO. Elle représente le comportement commun à tout itemObjet. Son diagramme ainsi que celui d'archipel.lang.ItemClass et de leurs relatives sont présentés à la figure 4.8.

A l'instar de la variable d'instance ISIT d'ObjVLisp, portée par chaque objet du modèle et qui prend pour valeur la classe de l'objet, ItemObject dispose d'une variable d'instance nommée caso qui référence le CASO de tout itemObjet. La présence de cette variable - et de son accesseur en lecture - se veut le pendant du getClass() de Java. Ainsi, tout itemObjet peut connaître sa classe réelle (getClass()) et sa classe vue comme un objet (getCaso()).

Toutefois, le comportement le plus significatif d'ItemObject est la mécanique de gestion des messages, vue comme le protocole "officiel" pour actionner un itemObjet (voir postu-

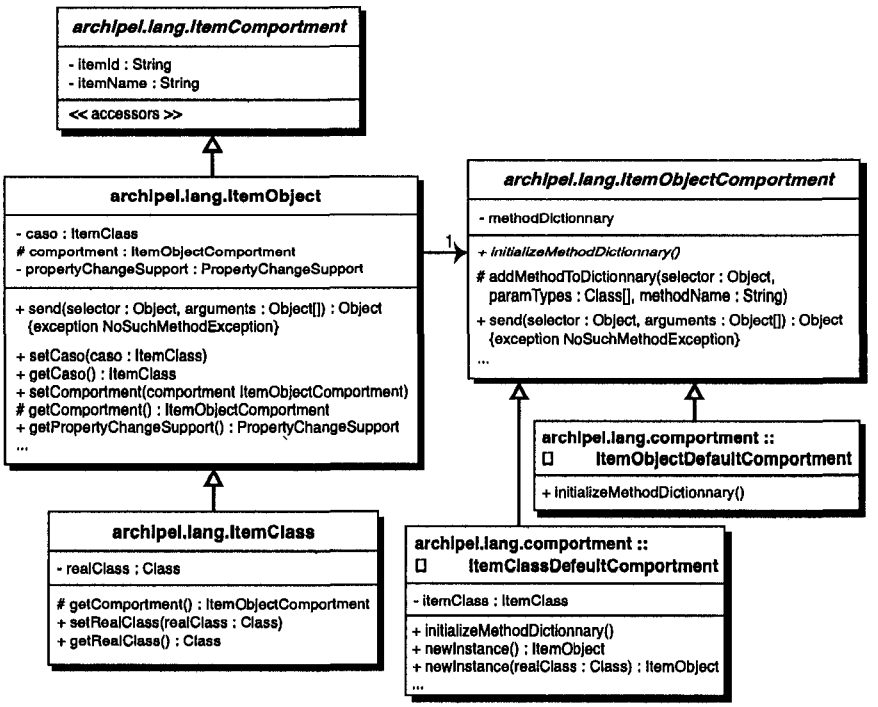

FIGURE 4.8 – Diagramme des classes ItemObject et ItemClass et de leurs relatives

lat ❷). Cette mécanique est supportée par la méthode send(selector : Object, arguments : Object[]) : Object {exception NoSuchMethodException}. Lorsqu'un message ne peut être compris, une exception du type java.lang.NoSuchMethodException est soulevée. Cette exception empruntée à l'API[12] Java doit être envisagée ici comme le DoesNotUnderstand de SmallTalk. L'algorithme de gestion des messages opère en deux temps :

1. le comportement de la variable d'instance comportment est invoqué. Du type archipel.lang.ItemObjectComportment, cette variable permet de déléguer à un objet externe la gestion de certains messages. Cette classe possède une méthode send(...) similaire à celle d'ItemObject. Si le message est compris, la valeur retournée par l'invocation de la méthode correspondante est retournée ;
2. si le message n'a pas été compris à l'étape 1 (exception soulevée), l'introspection est utilisée sur l'instance d'ItemObject ayant reçu le message. Si aucune méthode correspondante ne peut être invoquée, une exception est soulevée. Sinon la valeur de retour de l'invocation est retournée.

[12] *Application Programming Interface*, interface de programmation d'application

Cette mécanique de délégation permet d'ajouter à tout objet le comportement découlant de l'interprétation du contenu XML, en opposition au comportement explicitement décrit en XML. Ce dernier concerne les propriétés dont le rôle est de supporter la connaissance, et qui se transforment en attributs Java avec encapsulation par les méthodes d'accès adéquates, grâce aux stratégies d'écriture. A côté de cela, les comportements plus "algorithmiques", notamment liés à la gestion de la connaissance (par exemple, création d'instances), qui découlent du contenu XML sans y être explicitement décrits, sont délégués à un objet tiers. Cela permet en fait d'intégrer à tout itemObjet du code écrit en Java, dont la nature le rendrait difficile à décrire en XML. On ne parle par contre pas ici des traitements réalisés pour initialiser les valeurs des attributs, qui sont du ressort des stratégies d'initialisation. A l'inverse, il serait ridicule de devoir écrire à la main la multitude des classes nécessaires à la description d'un contexte matériel ou technologique, et dont la génération automatique ne requiert que très peu d'effort. C'est aussi un intérêt de ce langage et de ses outils d'édition graphique.

Le MOP de contrôle de l'envoi de messages Le modèle CASO correspond à l'introduction de méta-objets dans le langage Java. En servant d'interlocuteur pour l'utilisateur en lieu et place des classes Java traditionnelles par une mécanique unique de gestion des messages, il en permet le contrôle dans une approche traditionnelle en 2 temps : (1) recherche de la méthode correspondant au sélecteur (*lookup*), et (2) application de cette méthode. Les réifications sont celles du langage Java, soit essentiellement la classe java.lang.reflect.Method.

La classe ItemObjectComportment se présente comme un dictionnaire de méthodes, associant à un sélecteur une instance de java.lang.reflect.Method[13]. La recherche est donc effectuée au sein du dictionnaire sur le sélecteur. A noter que la sélecteur peut ne pas être le nom réel de la méthode, ce qui permet d'utiliser à ce titre des mots réservés du langage hôte. Si aucune méthode n'est trouvée, la recherche est complétée par la découverte par introspection des méthodes générées dynamiquement. Les mécanismes de réification propres au langage hôte sont alors mis en oeuvre. On remarquera qu'outre ces possibilités offertes par le *framework*, il est tout à fait envisageable de créer à la main des sous-classes d'ItemObject. Néanmoins, cette utilisation plus traditionnelle du langage ne correspond pas à la logique de représentation des connaissances proposée

[13]L'association peut être de type 1 à n, si le nombre et/ou le type des paramètres des méthodes sont différents.

jusqu'ici. Il est en effet plus simple de laisser le système créer la hiérarchie de classes et d'ajouter des comportements spécifiques là où cela est nécessaire. On se rappellera que l'association comportementale ItemObjet/ItemObjectComportment découle de la stratégie de chargement de l'item. Par défaut, l'ItemObjectComportment associé à tout ItemObject dispose d'une dictionnaire vide.

La classe archipel.lang.ItemClass Il existe un comportement dont la présence est évidente pour certains objets du modèle : celui de la création des instances. La classe ItemClass, qui hérite naturellement d'ItemObject (postulat ❺), propose comme comportement par défaut une instance d'ItemClassDefaultComportment, qui est elle-même une sous-classe d'ItemObjectComportment. Cet objet dispose d'une méthode de création d'instance, référencée dans le dictionnaire sous le sélecteur "new".

ItemClass, en tant que racine de l'arbre d'instanciation, propose donc naturellement les mécanismes nécessaires pour la création d'instances. Néanmoins, puisque ces mécanismes sont délégués, ils peuvent être modifiés par ses instances. Ce qui pourrait passer pour une incohérence est nécessaire de par la nature du langage Java. En effet, certaines classes Java n'ont pas la capacité d'être instanciées : il s'agit des interfaces. Un CASO ayant comme classe réelle une interface ne devrait donc pas comprendre le message new, ce qui est rendu possible grâce à cette mécanique de délégation. Cela permet du point de vue du modèle une représentation uniforme des CASO, quelque soit la nature de la classe réelle. Il en est de même en Java, les objets interfaces étant bien des instances de la métaclasse java.lang.Class, mais soulevant une exception si la méthode newInstance() : Object est invoquée[14].

Pour différencier instances terminales et classes dans ce modèle, la capacité à comprendre le message new n'est donc pas significative, contrairement à ObjVLisp. Ce qui l'est par contre, c'est la compréhension du message getRealClass, qui souligne l'existence d'une classe réelle. Cette propriété est portée par ItemClass, racine de l'arbre d'instanciation, qui détermine le comportement des itemObjets en tant que classes.

Détection de la modification de l'état d'un itemObjet La classe ItemObject possède un attribut de type java.beans.PropertyChangeSupport. L'objectif est d'offrir à l'échelle du langage une mécanique de notification (dans l'esprit du patron observateur [57]) lorsque les valeurs des variables d'instance d'un itemObjet sont modifiées.

[14]Voir documentation de l'API Java. http ://java.sun.com/reference/api/

Pour ce faire, le support de notification est automatiquement invoqué lorsqu'un message d'affectation est traité, dans le contexte du MOP de contrôle de l'envoi de messages.

Amorçage du modèle CASO L'amorçage vise à répondre au problème récurrent de régression infinie, ou plus précisément à celui de la création originelle dans le cas du modèle CASO. Dans ObjVLisp (postulat ❺), la racine de l'arbre d'instanciation, Class, est instance d'elle-même, posant un niveau fini de régression. L'amorçage est alors basé sur une première description "en dur" et minimaliste de Class, qui servira tout d'abord à créer la classe Object puis Class, sa redéfinition circulaire permettant la finalisation de la description de son comportement. Cette description en dur permet avant tout la mise en place d'une structure de représentation pour objets dans un langage hôte dont ce n'est pas la finalité première. Elle comprend de plus les méthodes new et initialize dont le corps est suffisant pour l'instanciation d'Object et l'auto-instanciation de Class, qui ne l'oublient pas hérite d'Object. La figure 4.9 remet ces deux classes en contexte.

Pour archipel.lang, la problématique est un peu différente. En effet, le langage hôte utilisé, Java, est un langage objet. Ainsi les classes archipel.lang.ItemObject et archipel.lang.ItemClass sont écrites en dur, ItemClass héritant d'ItemObject. L'amorçage vise alors à créer les CASO primaires du système, racines de l'arbre d'instanciation (le seul objet du modèle à devoir être créé en Java) et d'héritage. Le code est le suivant[15] :

```
import archipel.lang.ItemClass;
import archipel.lang.ItemObject;

/** Instantiation tree */
ItemClass itemClassCASO = new ItemClass();
itemClassCASO.setRealClass(ItemClass.class);

/** Inheritance tree */
ItemClass itemObjectCASO = (ItemClass) itemClassCASO.sendX("new", ItemObject.class);
```

Extension du modèle

La figure 4.9 illustre la similarité structurelle d'archipel.lang et d'ObjVLisp en ce qui concerne l'extension du modèle. Dans cet exemple, repris de [33], on voit que dans le modèle ObjVLisp l'introduction de la métaclasse MetaPoint impose que celle-ci soit instance de Class (c'est bien une classe) et qu'elle hérite de celle-ci (ses instances doivent

[15]La méthode sendX(...) traite l'envoi de messages lorsque la non compréhension est une erreur et non un exception.

pouvoir avoir le comportement de classes). Du côté archipel.lang, MetaPointCASO est instance d'ItemClassCASO - instance Java de sa classe réelle ItemClass -, et l'étend - sa classe réelle étend ItemClass -. Le comportement final reste identique à la logique d'ObjVLisp. Du côté de l'héritage, DefaultPoint étend Object côté ObjVLisp, alors que DefaultPointCASO dispose d'une classe réelle qui étend ItemObject. Jusqu'aux instances terminales, la logique est respectée.

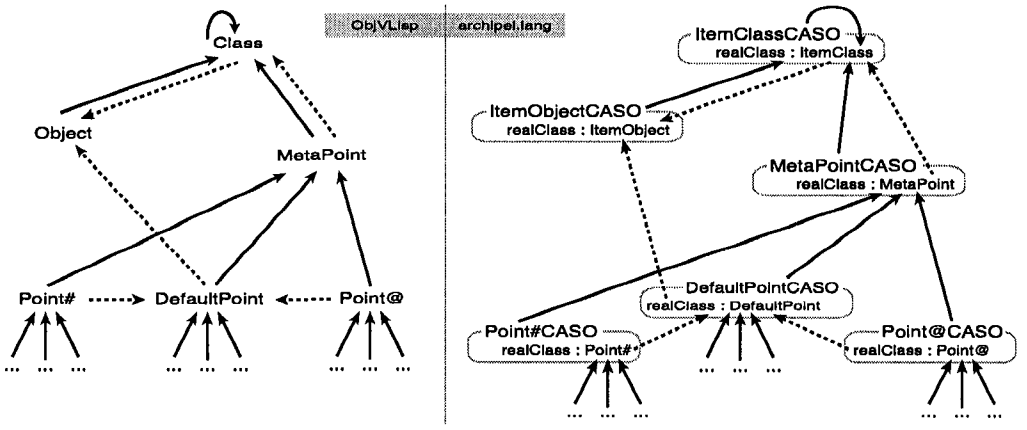

FIGURE 4.9 – Similarité des modèles ObjVLisp et CASO d'archipel.lang

4.3.4 Récapitulatif de la construction d'une représentation objet de la connaissance

L'ensemble de la mécanique du *framework* archipel.lang ayant été posée, il est possible d'en récapituler la logique de fonctionnement, en introduisant quelques aspects liés à la mise en oeuvre laissés jusqu'à présent de côté.

Le premier élément à prendre en compte est la classe ItemManager, présentée à la figure 4.10. Un ItemManager est associé à toute collection d'items décrite en XML, selon la mécanique des ressources du *framework* archipel.fwk.appmanager (voir annexe E). En complément de l'analyse syntaxique du contenu XML (voir annexe D) et de la création des objets du modèle structurel, ItemManager assure l'interprétation des propriétés des items. Pour ce faire, il est fait appel à la sémantique du langage, présentée sous la forme d'un objet instance de la classe LangSemantics, du package archipel.lang.bootstrap. Un tel objet dispose de méthodes permettant pour une propriété donnée, identifiée par son nom,

d'en connaître la stratégie d'interprétation, ainsi que tout ce qui concerne la manipulation graphique de l'information (agencements, actions, visiteurs, éditeurs de propriétés). Dès lors, la question est de savoir comment créer cet objet.

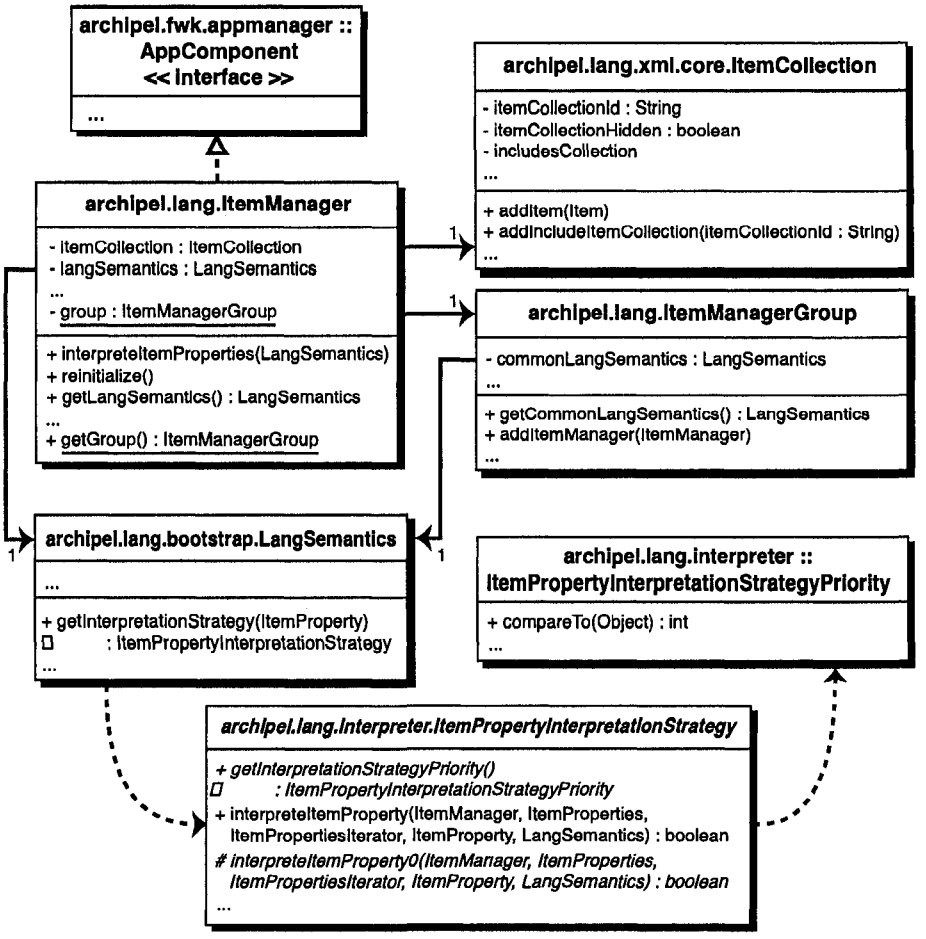

FIGURE 4.10 – Diagramme des classes dédiées à la gestion des collections d'items

Archipel est auto-décrit. Autrement dit, le contenu de l'objet LangSemantics est issu d'une collection d'items, qui doit elle même être interprétée. Pour ce faire, le système dispose de deux objets LangSemantics. Le premier donne la sémantique de la collection d'items décrivant la sémantique définie par l'utilisateur. Cet objet est instance de la classe BootstrapLangSemantics. Non présentée à la figure 4.10, cette classe hérite de LangSemantics et définit explicitement la sémantique nécessaire à la compréhension de celle définie

par l'utilisateur. Elle est codée "en dur", non modifiable et non extensible[16], répondant ainsi au problème de régression à l'infini posé par cette auto-description.

Le second objet LangSemantics, qui représente la sémantique définie par l'utilisateur, sera pour sa part qualifié de "sémantique usuelle", et correspond à la propriété commonLangSemantics portée par la classe ItemManagerGroup. Toutes les instances d'ItemManager font en effet partie d'un groupe, ce qui permet de les regrouper au sein d'une même entité, et notamment d'en faciliter la manipulation graphique[17]

La sémantique usuelle travaille à partir du modèle objet sémantique de la collection d'items décrivant la sémantique utilisateur. Ainsi, cette collection est automatiquement chargée sous forme d'ItemObjet. Ces derniers disposent d'un comportement adapté à leur fonction : par exemple, la classe des stratégies d'interprétation créé un index de ces stratégies, c'est-à-dire de ses instances, à partir du nom des propriétés. Elle est donc capable de répondre efficacement à des messages de sélecteur "getInterpretationStrategy" prenant comme argument le nom d'une propriété. Pour ce faire, une classe héritant d'ItemObjectComportment a été spécifiquement développée. Lorsque l'ItemObjet modélisant la classe des stratégies est chargé, une instance de cette classe lui est associée, conformément à la stratégie de chargement correspondante.

Pour illustrer concrètement cela, on reprendra l'exemple très simple de l'item "tasse bleue". A cet item est associée une propriété de nom "instanceof", dénotant son caractère d'instance dans le langage créé pour modéliser ce contexte matériel. Voici la méthode dédiée à l'interprétation de cette propriété, issue de sa stratégie d'interprétation. Cette stratégie est une sous-classe d'ItemPropertyInterpretationStrategy, elle-même présentée au sein du diagramme 4.10.

```
1 protected boolean interpreteItemProperty0(ItemManager manager,
      ItemProperties description, ItemPropertiesIterator iterator,
3     ItemProperty property, LangSemantics langSemantics) {

5   Item itemClass = manager.getItem(property.getPropertyValue());

7   if (itemClass == null) {
      manager.getLogger().warning(
9         "Instance of an unknown class or metaclass "
              + description.getItem().getItemId() + " -> "
```

[16]Du point de vue de l'éditeur graphique, cela se traduit par l'absence d'action permettant de créer de nouvelles classes, seules de nouvelles instances des classes décrivant les stratégies d'interprétation, les agencements, *etc.* pouvant être ajoutées.

[17]En somme, les systèmes d'onglet présents dans les copies d'écran de l'annexe F sont une vue du groupe d'itemManager, chaque onglet étant une vue d'un itemManager de ce groupe.

```
11              + property.getPropertyValue() + " ? ");
        property.setState(new ErrorIPS());
13      description.getItem().setState(new ErrorIS(description.getItem()));
        return false;
15    }

17    if (itemClass.getState().isItemStateUnknown()) {
        itemClass.interpreteItemProperties(manager, langSemantics);
19    }

21    // local property state.
      property.setState(new InstanceofIPS());
23
      // instanciation mechanism
25    itemClass.getState().getInstanciationStrategy()
          .createInstanciationRelation(itemClass, description.getItem(),
27            manager, langSemantics);
      return true;
29  }
```

Le premier traitement réalisé concerne la récupération auprès de l'ItemManager de la classe d'items dont "tasse bleue" est une instance (ligne 5). Cette requête est réalisée à partir de la valeur de la propriété "instanceof", qui est interprétée ici, puisque cette valeur ce veut être l'identifiant de la classe. Un test de nullité de la classe d'items est alors réalisé à la ligne 7. Dans l'affirmative, l'état de la propriété et de l'item "tasse bleue" sont mis à "erreur" et l'interprétation est arrêtée. A noter que l'éditeur graphique puise dans les états des Item et de leurs propriétés l'icône illustrant visuellement chaque donnée (voir diagramme de classes 4.6). Les états d'erreur se manifestent actuellement par une croix rouge, il est donc aisé de visualiser les difficultés d'interprétation du modèle, et d'aller en vérifier l'explication dans le fichier de trace. Si l'item existe, alors la stratégie s'assure que son interprétation a bien été effectuée (l.17-18). La propriété peut alors passer à l'état InstanceofIPS (IPS pour ItemPropertyState, l.22), puis la relation d'instanciation entre la classe d'items et le présent réalisée (l.25).

L'état InstanceofIPS fait de cette propriété une méta-propriété, dont la stratégie d'écriture est vide. Ce qui est beaucoup plus intéressant ici, c'est le résultat de l'application de la stratégie de création de la relation d'instanciation, dont voici le code. Rappelons que cette stratégie est associée à l'état de l'objet Item modélisant le concept de "tasse" dans notre exemple.

```
1 public void createInstanciationRelation(Item classItem, Item instanceItem,
      ItemManager manager, LangSemantics langSemantics) {
3
```

```
    // set the instanceItem state
5   instanceItem.setState(new InstanceofIS(instanceItem, classItem));

7   // add the properties to the instanceItem and modify their state
    for (ItemPropertiesIterator iterator = classItem.getItemDescription()
9       .iterator(); iterator.hasNext();) {

11    ItemProperty classProperty = iterator.next();

13    if (!classProperty.getState().isPropertyInstanciable()
          || !classProperty.isInstanceProperty())
15      continue;

17    if (!instanceItem.getItemDescription().hasItemProperty(
          classProperty)) {
19      try {
          instanceItem.getItemDescription().addItemProperty(
21            (ItemProperty) classProperty.clone());
        } catch (CloneNotSupportedException e) {
23        // supported
        }
25    }
      IPSFactory.setItemPropertyState(instanceItem.getItemDescription()
27        .getItemProperty(classProperty), langSemantics, false,
          false, true, instanceItem.getItemDescription(), manager);
29  }

31  // loading strategy
    instanceItem.getState().setLoadingStrategy(
33      new InstanceLoadingStrategy(classItem));
}
```

Le premier effet de cette stratégie est d'affecter à l'item *instance*, autrement dit "tasse bleue", un état dénotant son caractère d'instance (l.5). Jusqu'à présent, son état n'avait pas été modifié, il restait donc inconnu. En effet, cette charge ne peut incomber qu'à la classe d'items. Autrement-dit, si l'item modélisant le concept de "tasse" n'est pas interprété en tant que classe d'items, alors "tasse bleue" ne deviendra jamais instance.

Dans un second temps, la stratégie effectue le passage des propriétés de la classe vers l'instance, en clonant les objets ItemProperty. Le passage n'a lieu que pour certaines propriétés (l.13-14) : les "non-instanciables", comme les méta-propriétés sont éliminées. Il en est de même pour les propriétés ne représentant pas des variables d'instances. Tout cela est basé sur les réifications du MOP d'interprétation. De même, le passage n'est pas réalisé si l'item *instance* dispose déjà de cette propriété (l.17), sauvegardée avec sa valeur lors de la dernière persistance. Enfin l'état adéquat est associé à la propriété (l.26), grâce à la classe IPSFactory. Celle-ci s'assurera notamment qu'une stratégie d'interprétation

particulière n'est pas associée à cette nouvelle propriété, tout particulièrement pour en définir le type et assurer l'écriture du code adaptée.

Pour terminer, la stratégie de chargement InstanceLoadingStrategy est affectée à l'item *instance* (l.32). En voici le code.

```
public ItemObject loadItemObject(Item item, ItemObjectCache cache,
    ClassLoader classLoader) throws ItemObjectLoadingError,
    ItemObjectLoadingOrderException {

  // get the caso object
  if (!cache.containsItemObject(getClassItem()))
    throw new ItemObjectLoadingOrderException("No ItemObject "
        + getClassItem());

  // create the instance and set its common parameters
  ItemObject instance = (ItemObject) cache.getItemObject(getClassItem())
      .sendX("new");

  instance.setItemId(item.getItemId());
  instance.setItemName(item.getItemName());

  // add the instance to the cache
  cache.addItemObject(instance);

  return instance;
}
```

Deux remarques sont à faire ici. La première concerne la ligne 7. Le chargement nécessite d'avoir accès à l'ItemObject modélisant le concept de "tasse", pour pouvoir lui adresser le message de création. Mais rien ne garantit à ce stade que cet item à d'ores et déjà été chargé. C'est pourquoi une mécanique d'exception est utilisée pour garantir une logique dans l'ordre du chargement des items. Le présent sera mis de côté et son chargement tenté à la prochaine itération. Si jamais le nombre d'items ne pouvant être chargés ne diminue pas d'itération en itération, alors le système soulève une erreur.

La seconde remarque se trouve lignes 11 et 12, c'est l'envoi du message "new". On confirme ainsi que mis à part au niveau du bootstrap, l'ensemble des itemObjets créés dans le modèle le sont par envoi de message.

Pour terminer ce récapitulatif, quelques remarques sont nécessaires. Tout d'abord, l'interprétation des propriétés d'un item suit une logique, déterminée par la priorité des stratégies, cet ordre de priorité étant géré par un itérateur (patron de conception éponyme [57]). L'idée est que les stratégies (et donc les propriétés) les plus déterminantes pour

l'item soient interprétées en premier, tout simplement parce que ces stratégies risquent fort de modifier elle-mêmes l'état des autres propriétés (voir figure 4.5). C'est le cas des propriétés déterminant le caractère de classe de l'item, qui vont faire des autres propriétés de l'item qui ne disposent pas de stratégies spécifiques des variables d'instance. Leur état étant dorénavant connu, elles ne seront pas interprétées à nouveau. Mais cet état ne peut être connu que si la propriété qui fait de l'item une classe a été interprétée. Au final, cette mécanique simplifie l'algorithme d'interprétation, chaque propriété ne passant qu'une seule fois dans la boucle.

Pour finir, il existe dans cette mécanique bien d'autres classes qui n'ont pas été mentionnées. On citera notamment la classe BootstrapRoutine qui assure le chargement du modèle objet sémantique une fois les propriétés interprétées. C'est elle qui lance les procédures d'écriture des sources, de compilation, de chargement, d'initialisation, *etc.*, en faisant appel aux stratégies et autres outils appropriés. C'est donc elle qui est en charge de la création des racines des arbres d'héritage et d'instanciation du modèle CASO. A noter qu'une fois tous ces objets chargés, il sont placés dans un cache, accessible via un ItemObjetManager connu des itemManagers. On comprendra dès lors que si l'ensemble des collections d'items a été chargé sous leur forme objet sémantique, deux itemObjetManagers seront présents au sein de l'application. Le premier concernera la sémantique du langage, le second les éléments du langage. Cependant, du point de vue de l'utilisateur, le premier n'est pas directement accessible, car interfacé par l'objet représentant la sémantique usuelle, qui en cache l'existence. Il n'y a donc aucun risque de manipulation inappropriée de l'information[18].

4.3.5 Trois exemples d'utilisation du modèle

Au cours du récapitulatif qui vient d'être effectué, un exemple basé sur le modèle classe-instance a été mis en oeuvre. Dans cette section, trois exemples moins conventionnels sont rapidement présentés, afin d'ouvrir l'horizon des possibilités de représentation et surtout d'utilisation de la connaissance *via* archipel.lang.

[18]On notera aussi que tout ce qui concerne la sémantique du langage utilise un espace de nommage particulier, nommé "lang" (voir annexe C sur l'identification des items)

Classe-instance ou prototype ?

Lors de la description d'un contexte matériel, certaines questions pratiques se posent. Par exemple, si l'utilisateur dispose d'une machine à café, faut-il créer une classe "machine à café" puis une instance de cette classe alors que cette instance est unique et le restera certainement ?

Une solution pour ce cas de représentation particulier pourrait être la mise en oeuvre d'une logique de *prototype*. Un prototype est "une représentation typique d'une famille ou d'une catégorie d'objets" [51]. Les langages objets construits autour de cette notion, dits *langages à prototypes*, ne considèrent pas le concept de classe. Au contraire, l'idée est de "permettre une description simple des objets ne nécessitant pas la description préalable de modèles abstraits". Ainsi, dans ces langages, l'essence ne précède pas l'existence. La création des objets, d'un unique type, se fait - entre autre - *ex nihilo* et par clonage d'un objet existant.

Ce survol très incomplet des langages à prototype amène à ce poser la question suivante : que peut-on faire *via* archipel.lang pour décrire simplement un concept unique. Dans la mesure où la logique de classe ne peut être abandonnée, le plus simple est de faire du concept décrit une classe dont on créée automatiquement une instance (stratégie de chargement), et de cacher cette dernière à l'utilisateur. Pour ce faire, le MOP de contrôle de l'envoi des messages sera utilisé pour que ceux-ci soit transmis à l'instance alors que l'utilisateur se limite à interagir avec le concept unique qu'il a initialement décrit.

Patron de conception *Proxy*

Un des objectifs de cette mécanique de représentation est de permettre une manipulation complète et transparente des outils à disposition de l'orthèse pour la diffusion des actes d'assistance, à partir de l'entité représentant l'effecteur dans le modèle de connaissance. Pour ce faire, une solution parfaitement adaptée est l'introduction du patron de conception *Proxy*[19] au sein du modèle [57]. Le principe est simple : l'ItemObjet dispose d'une propriété, i.e. d'un objet, qui sert de mandataire ou disons d'interface avec l'effecteur réel. Cet objet mandataire pourra par exemple être issu d'une interface de programmation dédiée au contrôle logiciel de l'effecteur, ou jouer le rôle d'adaptateur pour cette interface.

[19]Terme anglo-saxon utilisé de manière courante en informatique. "Mandataire" serait une traduction française appropriée.

Dans tous les cas, ce qui importe ici c'est que les messages adressés à l'ItemObject soient relayés au mandataire lorsqu'il s'agit de contrôler l'effecteur. Le MOP de contrôle de l'envoi des messages rentre alors en jeu. Par introspection, il est possible de découvrir le comportement de l'objet mandataire, et de l'invoquer si celui-ci s'applique. Dans le cas contraire, le comportement est invoqué sur l'ItemObject. Le tout reste transparent car l'indirection peut être gérée au niveau de l'objet ItemObjectComportment associé à l'ItemObject. Celui-ci peut donc jouer un double rôle : décrire la connaissance selon la logique du modèle et servir d'intermédiaire pour son utilisation réelle. Le prochain chapitre, qui porte sur les interactions homme-machine, en présentera une illustration.

Raisonnement sur la connaissance décrite

Les raisonnements mis en oeuvre pour le présent prototype concernent essentiellement les interactions homme-machine. Leur description sera donc donnée au prochain chapitre, qui porte sur cette thématique. Leur faisabilité repose principalement sur le modèle CASO. Par exemple, le méta-objet associé au concept modélisant un service d'IHM pourra être utilisé pour garder la référence des classes d'effecteurs offrant ce service. La matérialisation de cette relation permettra un accès simple aux effecteurs, lorsqu'il sera question du service en question.

D'une manière générale, les raisonnements effectués actuellement au sein de l'application répondent avant-tout à une logique de requête, comme trouver un effecteur répondant à un ensemble de critères. Autrement dit, les raisonnements ne portent pas sur la création de nouvelles connaissances. Ainsi, même si inférence il doit y avoir dans le futur pour améliorer les capacités orthétiques de l'outil, l'essence des raisonnements n'est pas de ce registre. Cela légitime d'une certaine manière l'utilisation d'un modèle qui ne dispose pas d'une base formelle du niveau de certains autres langage de représentation.

4.4 Autres approches en représentation des connaissances

Diverses approches de la représentation des connaissances existent. Le langage généralement utilisé pour les applications sensibles au contexte est OWL (par exemple, [75–77]). Cette initiative du W3C (*World Wide Web Consortium*) se présente sous la forme d'un langage

à balises[20] permettant la définition d'ontologies structurées, initialement pensé pour le contenu *web* [152]. Ses fondements théoriques (tout particulièrement les logiques de description) et les outils développés pour son utilisation en font un langage séduisant. Sont accessibles aux développeurs des outils à source ouverte pour l'édition d'ontologie OWL (par ex., *Protégé* [56]) et la manipulation de manière programmatique du langage (par ex. *Jena* [1]), ainsi que divers moteurs d'inférence (par ex., *Pellet* [139]). Enfin, OWL n'est pas concerné par les limitations usuelles des langages de RC [28] : il ne manque pas de généralité car il est indépendant de tout domaine ; basé sur les logiques de description, il dispose d'une base formelle solide offrant consistance dans la représentation et effectivité du raisonnement.

De par la représentation et l'utilisation envisagée des connaissances contextuelles, OWL s'affiche comme un outils trop complexe. Sur la forme, son utilisation impose celle d'un outillage logiciel lourd et encombrant. Pour sa part, la solution actuelle permet aux connaissances de rester à l'intérieur de Java, en parfaite harmonie avec le langage hôte. Ainsi, la simplicité a été préférée aux trop larges capacités de raisonnement d'OWL. Mais l'expressivité n'est pas pour autant limitée. Le modèle CASO permet notamment une simulation satisfaisante du principe de métaclasse, dont l'utilité pour la représentation contextuelle est évidente. La mise en oeuvre reste sans complexité particulière, puisque directement basée sur la mécanique objet Java. Côté OWL, un tel niveau de représentation impose l'utilisation de la version la plus expressive du langage, baptisée OWL Full, qui pose des problèmes de décidabilité dans les raisonnements. Bien entendu, archipel.lang n'offre pas le support formel d'OWL. Mais on pourra conclure en disant qu'il reste parfaitement possible d'intégrer au sein du modèle, là où nécessaire uniquement, une mécanique de raisonnement basée sur OWL ou sur tout autre méthode formelle d'inférence.

Une autre approche bien connue de la représentation des connaissances est celle de la RCO, ou *représentation des connaissances par objets*. Un système de RCO a pour fonction "d'organiser et de gérer les connaissances autour de la notion d'objet et de fournir des mécanismes d'inférence destinés à compléter l'information disponible" [106]. L'essence des systèmes de RCO réside dans leur capacité à créer de la connaissance. En cela, ils sont très proches des langages basés sur les logiques de description tels que OWL, les techniques d'inférence variant (voir par exemple [106] pour une présentation de ces techniques). L'objectif est donc très éloigné des besoins d'Archipel, où les outils de représentation sont avant-tout vus comme des éléments dévoués à la programmation

[20] OWL est un dialecte XML basé sur un syntaxe RDF, ce qui n'est pas l'objet de la présente discussion.

contextuelle.

4.5 Conclusion et perspectives

De XML au modèle CASO, archipel.lang offre un support simple et ouvert pour la représentation et l'utilisation des connaissances, à des fins de raisonnement et de manipulation réelle des éléments contextuels décrits. Son adéquation avec les besoins spécifiques à l'assistance cognitive va être démontrée dans les prochains chapitres, là où l'utilisation de la connaissance peut être mise en situation. Cela sera tout d'abord le cas avec les problématiques d'interaction homme-machine. On y verra entre autre comment le langage permet la description non seulement de capteurs mais aussi des services qu'ils offrent pour la perception du contexte. Le tout permet de reconstituer les actions perceptibles de la personne dans l'environnement, qui serviront de base pour le monitorage des activités de la vie quotidienne. Du côté effecteur, la mise en place d'une mécanique du type *proxy* permettra l'utilisation transparente des dispositifs pour l'assistance.

En fait, à partir du moment où l'on accepte dans un contexte programmatique d'utiliser une mécanique d'envoi de message (concrètement une méthode "passe-partout" prenant comme paramètres le sélecteur et les arguments de l'appel) en lieu et place d'un appel "traditionnel" où le nom de la méthode est connu de tous, il devient possible d'utiliser archipel.lang pour bien des aspects. Cela suppose d'accepter de ne travailler qu'avec des instances d'ItemObject, dont le comportement reste parfaitement connu : réponse au message si compris ou exception soulevée. Du point de vue de la lisibilité du code, le sélecteur s'apparentant généralement au nom de la méthode, le problème n'est pas insurmontable. Du point de vue utilisabilité, la compréhension des messages ne peut être testée qu'à l'exécution, ce qui est bien peu par rapport aux environnements de développement actuels - tels qu'Eclipse[21] ou NetBeans[22] - qui effectuent en arrière plan un intégration continue du code. Il en est de même pour les problèmes de typage des objets retournés, l'utilisation de "cast" devenant indispensable. Le développement d'un plug-in Eclipse mettant à portée de main (disons d'un clic) la connaissance pourrait être un exercice intéressant. Du côté performance, c'est la compilation des sources Java qui est coûteuse en temps (linéaire par rapport au nombre de classes à compiler), réalisée

[21] http ://www.eclipse.org/
[22] http ://www.netbeans.org/

grâce à la librairie Tools de Java et dépendant clairement des capacités matérielles de la machine hôte. La préparation de la compilation (interprétation, écriture des sources, *etc.*) ne recèle aucune complexité algorithmique particulière. Il en est de même à l'utilisation, les mécanismes de réflexivité (introspection en l'occurrence) n'utilisant que la mécanique du langage Java. Des questions peuvent par contre être posées pour une utilisation dans un contexte distribuée, mais elles ne seront pas abordées ici. Au final, comme on le verra dans les prochains chapitres, l'utilisation d'archipel.lang au sein du *framework* est beaucoup plus large que ce pourquoi il a été initialement pensé.

Chapitre 5

archipel.hci

"W. Buxton[1] se plaît à imaginer comment un extraterrestre reconstituerait l'aspect physique d'un être humain s'il était en présence d'une de nos stations de travail comme seul objet relatant l'espèce humaine. Il imaginerait probablement un individu doté de deux membres, l'un pourvu d'un unique doigt, pour actionner la souris, et l'autre muni d'une centaine de doigts en cinq rangées pour utiliser le clavier. Son sens tactile serait pratiquement nul (les claviers sont peu sensibles à la pression). L'individu serait probablement un cyclope et son sens auditif juste suffisant pour reconnaître un "bip" entre 400 et 800 Hz..." ([19] cité dans [20], page 1).

Cette petite histoire "martienne" témoigne de tout une époque, non révolue, de l'informatique. L'*Interaction Homme-Machine* (IHM, en anglais HCI pour *Human-Computer Interaction*), est la "discipline qui s'intéresse au design, à l'évaluation et à la mise en oeuvre de systèmes informatiques interactifs destinés aux humains, ainsi qu'à l'étude des principaux phénomènes environnants" [68]. Avec l'arrivée des dispositifs de pointage, l'ordinateur est métaphoriquement devenu espace de travail, et a fini par prendre la place qu'on lui connaît aujourd'hui chez les particuliers et les entreprises. C'est loin d'être négligeable.

Et pourtant, il semble possible de faire mieux. C'est tout du moins ce que prônent les partisans de l'informatique ubiquitaire. Parce qu'il existe un décalage entre l'apport de l'informatique - immense -, et la façon dont l'information transite entre l'homme et la machine - inappropriée, limitée voire forcée -. "Sommes nous réellement intéressés à interagir avec des ordinateurs ? Notre objectif n'est-il pas plutôt d'interagir avec de

[1]Designer et chercheur, www.billbuxton.com

l'information, de communiquer et collaborer avec des personnes ? L'ordinateur ne devrait-il pas passer à l'arrière plan et disparaître ?" [141].

Les interfaces homme-machine, supports concrets (physiques ou numériques) de l'interaction, sont donc espérées en voie de disparition, tout du moins dans leur acception actuelle. Cette disparition pourrait être physique : grâce à la miniaturisation de la technologie, l'interface est intégrée aux objets ou aux vêtements. Elle peut aussi être mentale : l'artefact est bien là, mais il n'est pas perçu comme un ordinateur, ce qui semble être le cas avec les tables interactives [141]. Ainsi, les interfaces deviennent ambiantes, puisque c'est l'ensemble de l'environnement qui peut être média d'interaction [64]. L'utilisateur est alors délivré de ce que certains auteurs qualifient de "tyrannie de l'espace de travail", le traitement de l'information et la communication pouvant prendre forme n'importe où, n'importe quand, en prenant en compte l'ensemble des capacités sensorielles de la personne. En somme, l'interaction, plus naturelle et plus efficace, devient interaction "homme- environnement de traitement de l'information" [41].

Pour passer de la philosophie (dans le sens conception du rapport entre l'homme et la technologie) à la mise en oeuvre, les techniques de gestion de contexte ont déjà été présentées comme une étape nécessaire pour donner à l'informatique son caractère ubiquitaire (voir chapitre 2). Elles ont aussi été citées comme une approche pertinente pour la mise en place de systèmes d'assistance à la réalisation d'activités domiciliaires complexes. Celles-ci reposent tout particulièrement sur la notion d'IHM implicite, permettant au système d'aller, de façon non intrusive, au delà de ce que l'utilisateur exprime explicitement, pour adapter au mieux les services rendus. Plus précisément, une IHM implicite est "une action, réalisée par l'utilisateur, qui de prime abord ne revêt pas la forme d'une interaction avec le système informatique, mais que ce dernier comprend tout de même comme une entrée de donnée" [135]. Dès lors, il faut (1) être capable de percevoir cette action, et (2) pouvoir l'interpréter, pour la traduire en une donnée manipulable par le système.

Ce chapitre va dans un premier temps aborder la question de la perception des interactions implicites et leur interprétation sous la forme d'événements atomiques, l'interprétation de plus haut niveau (le monitorage des activités) faisant l'objet du chapitre suivant. Puis, dans un second temps, c'est le sens machine vers homme de l'interaction qui sera présenté. On verra comment, dans Archipel, l'ensemble de l'environnement est sujet à supporter la diffusion d'informations d'assistance, dans un esprit qui n'est pas réellement celui de la disparition de l'informatique, mais plutôt de sa relocalisation dans

les objets du quotidien. Cette séparation traditionnelle des "entrées" et des "sorties", artificielle mais simplificatrice, sera dans un troisième temps regardée à un niveau plus macro. L'ensemble - indissociable - forme en effet la base du processus de communication entre l'homme et la machine, les messages concernant l'assistance cognitive.

5.1 Perception et interprétation des IHM implicites

L'objectif affiché ici est de fournir au système d'assistance les outils nécessaires pour la perception et l'interprétation de bas niveau des actions de la personne, afin de reconstituer les événements dénotant le contexte d'assistance cognitive. Tel que présenté au chapitre 2, on aborde ici la question de la perception *externe*, c'est-à-dire celle des événements dont l'orthèse n'est pas à l'origine. Plus précisément, ces événements vont être principalement causés par le client lors de manipulation des artefacts matériels.

A la base de la perception se trouve la connaissance qu'a le système de l'environnement au sein duquel évolue la personne, soit le cadre de réalisation des AVQ. Celui-ci est jalonné de capteurs et d'effecteurs fournissant des services d'interaction homme-machine, ainsi que d'un ensemble d'artefacts jouant une rôle au sein de cette réalisation. Archipel.lang va alors être utilisé pour répondre à cette problématique de représentation et d'utilisation des connaissances.

5.1.1 Représenter et utiliser le cadre de réalisation des AVQ

Avec archipel.lang, l'idée est de disposer d'un support intégré pour la représentation de la connaissance et son utilisation dans un contexte programmatique, ici celui de la perception. Quelques travaux ont été réalisés sur la représentation, notamment pour les habitats intelligents dédiés aux personnes en perte d'autonomie cognitive [88]. Sur le fond, les modèles proposés, décrits en OWL, s'inscrivent pleinement dans le cadre du contexte d'assistance cognitive et seraient ainsi susceptibles de répondre aux besoins d'Archipel. La question de la forme, qui ne s'inscrit pas dans la logique de développement du *framework*, a été évoquée au chapitre 4. A côté de cela, il existe plusieurs *frameworks* dédiés à la gestion du contexte, offrant une mécanique de base pour l'acquisition des données contextuelles, les traitements de plus haut niveau (soit l'utilisation du contexte) étant laissés à la charge de l'utilisateur. *Context ToolKit* [48] et *JCAF* (Java Context-Awarenesse Framework) [10] répondent à ce profil.

Dans Context ToolKit, chaque capteur est associé à une entité logicielle appelée *widget*, qui fournit un premier niveau d'abstraction contextuel en transformant la donnée brute du capteur en une donnée manipulable par un programme. Les données contextuelles non issues de capteurs (comme le temps) sont aussi représentées par un *widget*. Un second niveau est fourni par les *interpreter*, chargés d'agréger ou d'abstraire l'information contextuelle provenant d'un ou de plusieurs *widgets*. Enfin, les entités de type *server* servent d'interface entre l'application et les entités contextuelles. Une mécanique d'événements permet à l'application d'être avisée des changements de contexte. Les différents composants peuvent être répartis sur un réseau, la communication étant assurée par envoi de messages XML sur TCP/IP. Une API[2] Java mélangeant coeur de l'application et implémentations diverses est *a priori* disponible mais non maintenue (elle date de 2000), la documentation laissant supposer que la mise en oeuvre de ces composants reste complexe. On peut néanmoins penser que l'aspect communication, mise en oeuvre de façon *ad hoc*, représente une large portion de l'API. ContextToolKit a été utilisé dans plusieurs prototypes, dont le "Conference Assistant" [47]. Cet assistant permet aux participants d'une conférence, qui disposent de l'application "cliente" sur leur PDA, d'avoir sous la main les informations contextuelles liées à la présentation à laquelle ils assistent. Il peut par exemple s'agir d'informations concernant le présentateur (nom, affiliation académique, site internet et adresse de courriel, *etc.*), ou la présentation, comme le numéro du transparent en cours de projection. A partir de là, le participant peut prendre des notes contextualisées sur la présentation, envoyer ses questions "asynchrones" par courriel au présentateur, *etc.*

Le *framework* JCAF [10] semble pour sa part beaucoup plus simple d'approche. L'API, compacte, représente un domaine autour des éléments suivants :

- les *entités*, qui permettent de représenter les concepts du domaine ;
- le concept de *contexte* qui, associé à une entité, en dénote le contexte ;
- les *éléments de contexte*, qui décrivent le contexte d'une entité ;
- les *relations*, chargées de spécifier le lien entre une entité et un élément de contexte.

Une mécanique traditionnelle de notification d'événements (patron observateur [57]) est utilisée pour le suivi des changements de contexte. Ces changements sont provoqués par des *clients de contexte*, qui forment une couche d'abstraction pour les capteurs et autres sources de données contextuelles. Les objets peuvent être distribués selon la mécanique

[2] *Application Programming Interface*, interface de programmation d'application

RMI[3], incorporée au modèle. Cette mécanique est aussi utilisée pour permettre à une application de découvrir les entités distribuées. Enfin, JCAF utilise l'API de sécurité Java pour valider l'accès des clients de contexte aux éléments des entités. En somme, l'API JCAF, qui date de 2004, répond nettement plus à la logique de la plateforme Java que Context ToolKit, l'auteur souhaitant faire de son travail un "standard" Java pour la gestion de contexte [10]. JCAF a notamment été utilisée pour le développement d'un lit d'hôpital sensible au contexte [11], assurant par exemple l'affichage sur écran des données médicales en fonction du patient et du personnel soignant présent autour du lit, le contexte étant défini par une technologie quelconque d'identification.

Ces deux *frameworks* adoptent l'approche maintenant traditionnelle en trois étapes des architectures sensibles au contexte pour la perception de ce dernier, et dont Context ToolKit fut l'initiateur [31] : (1) capturer l'information de bas niveau, (2) interpréter cette information en une donnée de plus haut niveau utile pour l'application, et (3) fournir l'information de haut niveau à l'application, en l'occurrence par une mécanique événementielle. Archipel n'échappe pas à la règle. Néanmoins, la mise en oeuvre va être simplifiée car intimement liée à la représentation des connaissances. La mécanique d'interprétation fournie par le *framework* va en effet se greffer sur le modèle de connaissance.

Dans Archipel, l'information implicite de haut niveau correspond aux événements atomiques résultant des actions du client dans l'environnement. Il peut s'agir par exemple de la mise en marche d'un appareil électroménager. Ces actions représentent autant de nouvelles situations contextuelles que le système va chercher à replacer dans le processus de réalisation de l'activité : l'utilisation en cours de l'appareil électroménager correspond-t-elle à un cheminement correct pour la présente activité ? Les événements qui les dénotent constituent donc des éléments clé de l'application, qu'elle doit pouvoir représenter et manipuler. Concrètement, les événements d'interaction sont à la base du monitorage et a fortiori de la représentation des AVQ, comme cela sera illustré au prochain chapitre.

En tant qu'information interprétée, les événements dépendent des dispositifs technologiques de plus bas niveau présents dans l'environnement. Si la personne ouvre la porte du réfrigérateur (action), alors l'événement d'ouverture correspondant, qui dénote la modification contextuelle, ne pourra être pris en compte *implicitement* si aucun dispositif n'en permet la captation[4]. A partir de là, l'idée est d'être capable, étant donnée la des-

[3] *Remote Method Invocation*, API Java permettant l'invocation de méthodes sur des objets distants.

[4] Lorsque le système "sait" qu'il n'est pas capable de capter implicitement une action (le contexte n'est que partiellement observable), il pourrait suggérer à la personne qu'elle le notifie explicitement de

cription des dispositifs bas niveau et tout particulièrement de leurs *services d'IHM*, de générer un ensemble d'objets représentant les événements possibles - rendus ainsi manipulables pour la représentation et le monitorage - et d'associer à ces objets la chaîne de traitement adéquate pour assurer la perception, l'interprétation bas niveau et la communication de l'effectivité de la réalisation de ces événements. C'est ce qu'illustre la figure 5.1, dont voici la lecture, présentée colonne par colonne.

Environnement physique

La figure 5.1 présente l'exemple d'un mélangeur dont les arrivées d'eau (chaude et froide) sont équipées de débitmètres. Ces derniers sont connectés de façon filaire à un contrôleur, en l'occurrence un PLC[5] (*Programmable Logic Controller*). Ce type d'équipement possède une unité de traitement de base avec mémoire gérant les rapports entre les dispositifs connectés en entrée et ceux en sortie grâce à un langage d'automatisation. Les PLC sont généralement utilisés pour le contrôle de processus industriels, comme les chaînes de montage. Dans le cas présent, le PLC dispose d'un compteur d'impulsion (un débitmètre n'étant qu'une roue dont la vitesse de rotation dénote celle de l'écoulement), dont il traduit la fréquence en une chaîne de caractères, autrement dit dans un formalisme interprétable par un programme informatique. Cette information est alors mise à disposition d'une application grâce à un module de communication réseau ethernet.

Représentation des connaissances

Pour décrire ces concepts, archipel.lang est utilisé sous la forme d'un langage à métaclasses complet avec héritage simple, pour rester conforme au langage hôte. La relation d'implémentation est utilisée pour décrire des relations sémantiques autres que l'héritage ou l'instanciation. Certains ItemObject du modèle vont donc avoir pour classe réelle une interface Java. Sont décrits ici :

- **Le contexte matériel**, illustré par un extrait de la hiérarchie des équipements domestiques, avec pour instances les deux robinets ;
- **Les services d'IHM**, qui décrivent la capacité des dispositifs technologiques de type capteurs à témoigner des modifications de l'état de l'environnement et de ses

cette réalisation. Cette approche sera discutée ultérieurement.

[5]En français, API pour *Automate Programmable Industriel*

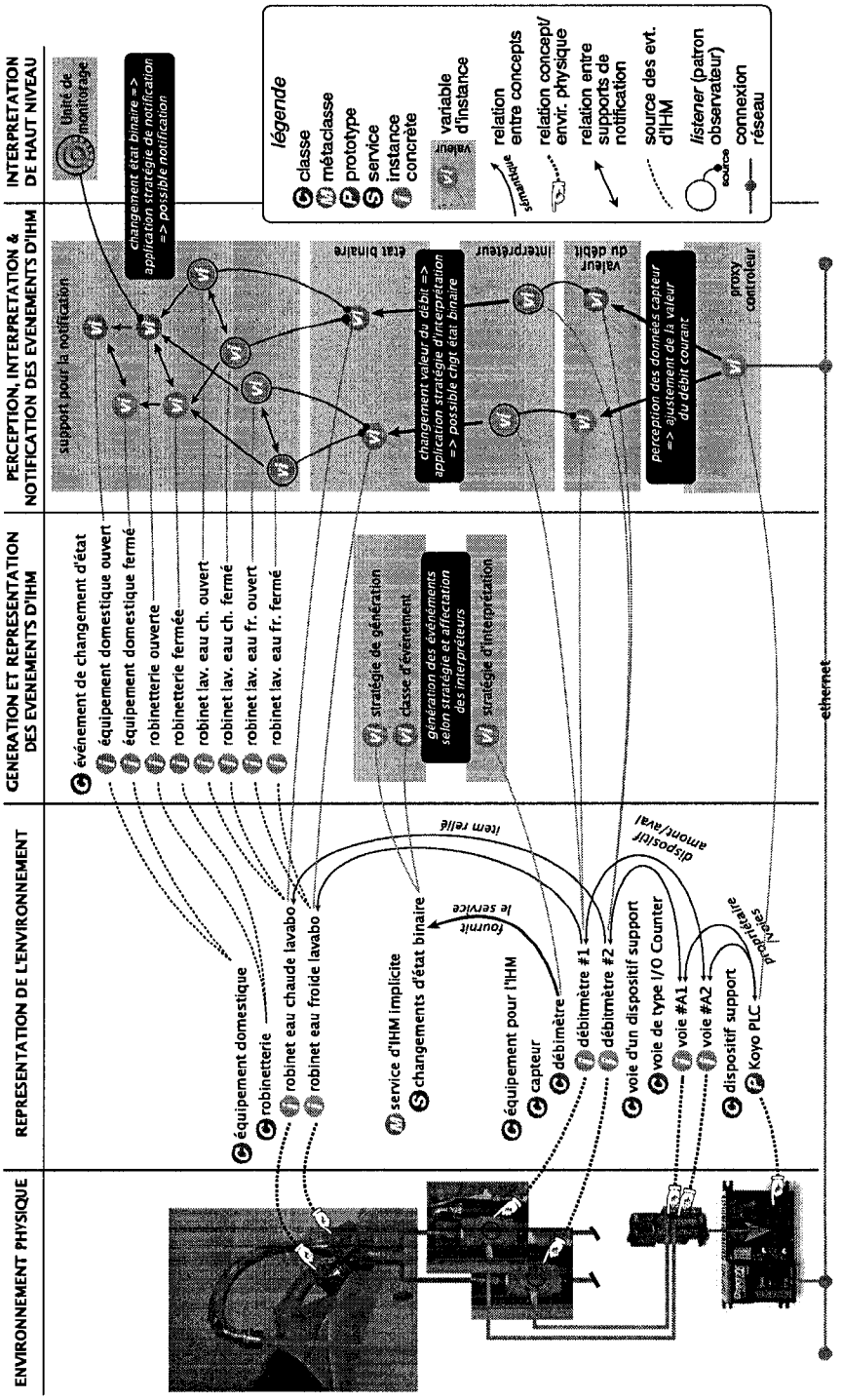

FIGURE 5.1 – De l'environnement physique à l'interprétation haut niveau des événements d'IHM implicites

composants (artefacts et humains). L'exemple est celui d'un service de captation des changements d'état binaire. Les services sont représentés comme des concepts à part entière, que pourront fournir certains capteurs. Cette relation sera concrètement exprimée par un lien d'implémentation.

- **Le contexte technologique**, avec un extrait de la hiérarchie des équipements pour l'interaction homme-machine. On y retrouve des classes de capteurs, notamment les débitmètres qui fournissent le service sus-nommé. La classe des dispositifs support (i.e. des contrôleurs) est présente, avec comme sous-classe un prototype de PLC de marque Koyo. Enfin les voies, c'est-à-dire les entrées ou les sorties d'un dispositif support, sont décrites. Plus précisément, sont présentes deux voies de type *I/O Counter*, autrement dit des compteurs d'impulsions. A noter que les services d'IHM font partie du contexte technologique.

La figure 5.1 illustre également les relations qui existent entre ces concepts, et qui témoignent des relations physiques présentes dans l'environnement. Ainsi le PLC Koyo dispose de différentes voies, sur lesquelles sont connectées les débitmètres. Ces derniers sont associés à des artefacts particuliers, en l'occurrence des robinets dont ils "observent" l'alimentation en eau. L'ensemble va permettre d'assurer l'interprétation de bas niveau de l'information (du contrôleur aux robinets), à partir d'une représentation simple et naturelle. Un éditeur *ad hoc*, offrant une vue adaptée du modèle objet sémantique, a d'ailleurs été développé pour permettre de spécifier graphiquement les relations physiques existantes entre ces dispositifs.

Génération et représentation des événements d'IHM

Principe général Les services d'IHM constituent le point d'entrée du processus de génération. D'un point de vue algorithmique, ce processus peut être résumé comme suit :

début
pour chaque service d'IHM présent dans l'environnement **faire**
  pour chaque capteur de l'environnement proposant ce service d'IHM **faire**
    générer les événements correspondant à la sémantique du service d'IHM pour l'artefact concerné par ce capteur
    pour chaque sur-type de l'artefact concerné par ce capteur **faire**
      générer les événements pour le sur-type correspondant à la sémantique du service d'IHM

fin pour
fin pour
fin pour
fin

Stratégies de génération des événements d'IHM La génération se fait sur la base de stratégies, c'est-à-dire autour d'une famille d'algorithmes de génération, dans l'esprit du patron de conception éponyme [57]. Chaque service spécifie la stratégie qui lui est associée, *via* une variable d'instance donnant le nom de la classe Java de la stratégie. Cette variable d'instance est logiquement définie au niveau de la classe des services, soit la métaclasse "service IHM implicite". L'application des stratégies est réalisée par le *moteur d'IHM*. Ce processus autonome, issu du sous-*framework* archipel.hci, a pour rôle principal la gestion des requêtes d'IHM, comme cela sera précisé dans la seconde partie de ce chapitre. Il est construit comme un AppComponent, autrement-dit comme un composant d'application qui peut être dynamiquement chargé, rechargé, sauvegardé et arrêté, tel que le spécifie le *framework* de gestion d'application archipel.fwk.appmanager (voir annexe E). C'est lors du chargement de ce moteur que l'application des stratégies est réalisée.

De l'application découle la création des événements d'IHM dont la classe, décrite en archipel.lang, est précisée par le service. Pour l'exemple de la figure 5.1, il s'agit de la classe des événements de changement d'état. Conformément au principe algorithmique présenté ci-dessus, la génération se fait pour chaque capteur de chaque service. Cela est rendu possible par la spécificité comportementale des classes manipulées, qui gardent une référence sur leurs instances, sous-classes (si relation d'héritage) ou classes implémentant (si relation d'implémentation). Les références sont stockées au niveau des méta-objets (modèle CASO), l'affectation d'une référence étant déterminée par les stratégies d'initialisation des propriétés des ItemObject (MOP d'interprétation des propriétés, voir chapitre 4). Dans le cas présent, la métaclasse "service d'IHM implicite" va connaître ses instances, soient les différents services d'IHM. Ces derniers vont pour leur part référencer les classes des capteurs fournissant le service, qui vont elle-mêmes connaître leurs instances et/ou sous-classes, *etc.* En résumé, le moteur d'IHM n'a à connaître explicitement qu'un seul item : la métaclasse "service d'IHM implicite", point d'entrée de cette mécanique de raisonnement. L'ensemble est donc parfaitement adaptable et extensible, puisque l'arrivée d'un nouveau service, type de capteur ou capteur concret ne changera en rien l'algorithme

sous-jacent.

La génération se fait aussi pour les abstractions conceptuelles des items associés aux capteurs[6], donnant ainsi naissances aux huit instances d'événements (soit deux robinets × deux niveaux d'abstraction pour une thématique de changement binaire). De cette génération découle la possibilité d'abstraire les raisonnements côté compréhension du CAC, c'est-à-dire lors du monitorage des activités. L'ouverture du robinet d'eau chaude pourra ainsi être interprétée au même titre que celui d'eau froide, si la distinction n'a pas lieu d'être. L'événement monitoré sera celui de l'ouverture d'un robinet, quel qu'il soit, ce qui est non seulement logique mais plus simple à modéliser qu'un *ou* entre deux événements (le robinet d'eau froide *ou* d'eau chaude).

Association capteurs/service d'IHM La génération donne aussi lieu à l'affectation pour chaque capteur de son *interpréteur*. Cet objet, qui se présente sous la forme d'un patron de conception stratégie [57], doit permettre au système de répercuter un changement côté capteur sur l'artefact auquel est relié le capteur. Les événements d'IHM témoignent en effet de la modification d'une caractéristique d'un artefact.

La classe de l'interpréteur va dépendre du type de capteur (la métaclasse correspondante n'est pas illustrée). Pour comprendre cela, il faut préciser que l'établissement d'une association entre une classe de capteurs et un service d'IHM repose sur deux critères : (1) les capacités techniques du capteur, et (2) les besoins de l'application.

Un contact électromagnétique ne peut renseigner que sur l'état binaire d'un artefact, une porte par exemple, que l'on peut savoir après interprétation ouverte ou fermée. Le service d'IHM qu'il fournira sera donc limité à la captation des changements d'état binaire. Un débitmètre, par contre, fournit la valeur du débit de l'eau sur une canalisation. Deux types de services peuvent alors être envisagés, l'artefact considéré ici étant un robinet. Soit l'intérêt au niveau applicatif est de savoir si le robinet est ouvert ou fermé, une valeur de débit servant de seuil de passage d'un état binaire à l'autre. Le service d'IHM est alors le même que pour le contact électromagnétique. Soit l'objectif est de connaître les variations du débit pour calculer par exemple la quantité d'eau utilisée. Alors le service d'IHM concerne les changements de valeur. En somme, pour un même service d'IHM, différentes *stratégies d'interprétation* peuvent être nécessaires, celles-ci dépendant du type de capteur (ou éventuellement du capteur si l'on veut raffiner le

[6]On notera qu'un item n'a aucune connaissance du ou des dispositifs de captation le concernant. La relation est spécifiée côté capteur : c'est sa raison d'être.

raisonnement).

Perception, interprétation et notification

Une fois la phase de génération passée, voici quelle est la chaîne de raisonnement s'appliquant à l'éxecution. L'ItemObject Koyo PLC dispose d'une propriété qui officie à titre d'interface entre celui-ci et le réseau ethernet. C'est son *proxy*, dans l'esprit du patron de conception éponyme [57]. Cet objet dispose des connaissances suffisantes pour dialoguer avec le PLC réel, et donc être avisé des changements de valeur d'un débitmètre. A noter que l'établissement et la coupure de la connexion réseau entre l'objet et le dispositif réel sera gérée selon le cycle de vie du moteur d'IHM, conformément à son rôle d'AppComponent (voir section précédente), tous les dispositifs supports utilisant une même interface de programmation.

Lorsqu'une chaîne de caractères est émise par le PLC réel puis reçue par le *proxy*, elle est tout d'abord interprétée selon l'organisation physique des capteurs, décrite dans le modèle de connaissance. On remonte ainsi du PLC vers une de ses voies puis le capteur qui y est connecté. La nouvelle valeur de débit est alors affectée à l'ItemObject capteur, par envoi de message. L'interpréteur du débitmètre, qui s'est enregistré comme observateur (*listener*) de l'attribut "débit" est notifié de ce changement. Cette mise en oeuvre du patron de conception observateur repose sur les possibilités offertes en ce sens par le modèle CASO et le MOP de contrôle de l'envoi des messages (voir chapitre 4, section 4.3.3). L'interprétation va alors avoir comme conséquence possible le changement de l'état binaire du robinet.

Stratégies de notification Cette propriété va être elle même écoutée par les objets supports pour la notification des événements d'IHM, auprès desquels peuvent s'enregistrer les outils chargés de l'interprétation de haut niveau. C'est le cas des unités de monitorages, dédiées au suivi de la réalisation des activités (voir chapitre 6).

La manière dont la notification est réalisée, on parlera à nouveau du patron de conception stratégie, dépend du service d'IHM. Dans l'exemple de la porte et du robinet, c'est un changement d'état sur un artefact à état binaire manipulable par la personne qui est attendu[7]. Lors de leur initialisation, à la génération des événements d'IHM, les supports

[7]La nature des changements pouvant intéresser l'application est variée : changement d'état, si binaire interprétable sous divers angles - allumage/extinction, ouverture/fermeture, utilisation/non utilisation, présence/absence de présence, activité/pas d'activité, *etc.* -, mais aussi changement de valeur, de position (objets, personnes), de contenu, *etc.*

pour la notification sont avisés de leur support père ainsi que de leur complémentaire. La connaissance du père permet de répondre au problème d'abstraction du raisonnement en matière d'IHM, cité plus haut. On peut ainsi aviser du changement d'état binaire pour l'ensemble de la hiérarchie d'équipement domestique.

La connaissance du complémentaire lorsque cela a du sens, ce qui est le cas pour des changements binaires, permet d'aborder la question la remise en l'état de l'environnement. Si l'application souhaite connaître l'utilisation de l'eau, et être donc notifiée de l'ouverture d'un robinet quel qu'il soit, veut-elle pour autant être avisée de la fermeture du robinet ? Dans certains cas oui, cela peut être l'objet même du monitorage : s'assurer que le bain ne débordera pas, problématique réelle pour la clientèle avec troubles cognitifs. Mais dans le cas contraire, que devient l'événement de fermeture du robinet ? Du point de vue du monitorage des événements, il est *orphelin*, car non attendu (seul l'utilisation qui se concrétise par une ouverture importe). Pourtant, cet événement est tout à fait logique *du point de vue humain*, car il correspond à la remise en "état de repos" d'un dispositif de l'environnement. Vu sous l'angle de son complémentaire, il n'est donc pas réellement orphelin. Or la prise en compte des événements réellement orphelins est impérative pour le système, car ils peuvent dénoter un comportement non adapté à la situation. Dès lors, on considérera que les événements de remise en l'état orphelins, complémentaires d'un événement non orphelin, ne doivent pas être générés. Cela revient à avoir une vision un peu plus macro des actions de la personne, sans aller à ce niveau du raisonnement jusqu'au concept d'activité.

Interprétation de haut niveau

On ne dira que très peu de choses de l'interprétation de haut niveau dans ce chapitre, le suivant s'intéressant exclusivement à la question. Comme cela a déjà été dit, les entités logicielles chargées du monitorage peuvent se placer comme observateur des supports de notification. Plus précisément, on verra comment cette association prend forme, étant donnée la modélisation de l'activité dont la réalisation doit être suivie.

Au final, la modélisation offerte répond efficacement à la problématique de perception, interprétation et notification des interactions implicites pour un environnement donné. Elle est par contre contraignante, car nécessitant une description complète de cet environnement. Néanmoins, ce passage obligatoire par la représentation des connaissances fait aussi la force de cette approche. En effet, elle donne une dimension complète à la

notion de gestion de contexte : il va y avoir une interprétation haut niveau menant éventuellement à la production d'une assistance cognitive, mais il y a déjà au niveau de la perception la prise en compte du contexte technologique (services d'IHM fournis par les capteurs) et de sa relation avec le contexte matériel pour la génération des événements d'IHM, leur interprétation et notification. Ce premier niveau de gestion de contexte offre au *framework* une modularité d'une grande pertinence. Enfin, cette mécanique facilite le développement d'applications, car la simulation de toute la chaîne de traitement des interactions implicites ne tient qu'à un simple envoi de message.

5.1.2 Le sort des interactions explicites

Avec les interactions implicites, on ouvre la porte à la disparition de l'ordinateur tel qu'on le connaît. Ainsi, dans l'absolu, la personne pourrait vaquer à ses occupations et être aidée en cela par un système "invisible", qui ferait des actions de la personne sa source d'information en entrée, sans que celle-ci ait à modifier la réalisation des dites-actions. Si elle était dès à présent envisageable, cette solution serait-elle pour autant viable ?

Il est clair que les interactions implicites viennent enrichir la relation entre l'homme et la machine. Pour autant, il reste délicat d'envisager la disparition totale de l'ordinateur : comment les personnes pourront-elles accepter de vivre avec un système qui, bien que parfaitement utile et dévoué à l'utilisateur, ne semble avoir aucune consistance ? Les interactions implicites enrichissent l'IHM car elles ouvrent la voie à la prise en charge du non-dit, dans l'esprit des communications humain à humain. Dès lors, comment appréhender du point de vue technologique, dans cette démarche mimétique, la part d'explicite des échanges entre hommes, sans revenir à la solution "traditionnelle" de l'ordinateur personnel que l'on sait inadaptée ? La solution est en principe simple : trouver de nouveaux ponts entre le monde informationnel et le monde physique, celui où vivent les humains. Une démarche très intéressante en ce sens est celle des *interfaces saisissables* [53]. L'idée est de disposer d'objets physiques manipulables, chaque objet étant associé à une fonctionnalité particulière du système[8]. Si l'utilisateur peut établir une relation implicite entre l'objet et la fonctionnalité, guidé en cela par la forme et/ou

[8]Les interfaces saisissables permettent un *multiplexage spatial* de l'interaction : chaque fonctionnalité devient utilisable de façon indépendante et simultanée. Elles s'opposent au *multiplexage temporel* des dispositifs de pointage traditionnels, où un dispositif d'entrée "générique" et physiquement neutre passe au cours du temps d'une fonctionnalité à une autre.

l'usage de l'objet (cette relation étant qualifiée par les spécialistes d'*affordance*), alors l'apprentissage et a fortiori l'utilisation du système devraient être facilités. Par exemple, un souvenir peut servir de signet physique pour un album photographique numérique : en posant un palmier miniature sur un écran, on accède aux photographies des dernières vacances dans les îles [93].

Dans une démarche d'assistance cognitive, l'accès physique et explicite au système reste indispensable. La personne doit pouvoir rester maître de l'orthèse et "l'arrêter" si elle en ressent le besoin, même si certains services purement liés à la sécurité des individus peuvent échapper à cette règle. Elle doit pouvoir aussi demander explicitement de l'assistance au système. Du côté application, il ne faut pas non plus, en l'état actuel des recherches, négliger la possibilité de demander à la personne de confirmer certaines inférences "problématiques". C'est le cas dans Archipel pour le monitorage des activités, lorsque le manque d'informations implicites ne permet pas au système de mener à bien ses réflexions. S'agissant de la matérialisation des interfaces, aucun travail n'a été mené dans ce sens au cours de cette thèse, même si le *framework* permet la réalisation de prototypes simple, du type de celui cité en exemple. Concernant l'utilisabilité présumée des interfaces saisissables avec la clientèle présentant des troubles cognitifs, la question, très pertinente, reste ouverte.

Du point de vue de la représentation, les interactions explicites collent à la mécanique de notification des implicites, dans le but d'uniformiser le traitement au niveau applicatif. Néanmoins, la comparaison s'arrête là : il n'existe pas de génération a priori ni de hiérarchie de notification. Les objets représentant les interactions explicites sont créés au fur et à mesure des besoins par le moteur d'interaction, à la demande des fonctionnalités concernées. Enfin, on notera que dans Archipel l'explicite devient en quelque sorte la solution par défaut, lorsqu'aucune interaction implicite ne semble pouvoir renseigner le système. Cet aspect sera précisé par la suite.

5.2 De la machine à l'homme

L'iceberg de l'assistance cognitive présente, en sa partie immergée, ce que l'on a appelé les *actes d'assistance*. Dans bien des cas, un acte d'assistance va se concrétiser pour une interaction machine vers homme. Si le système est par contre amené à intervenir directement sur les artefacts de l'environnement, pour des raisons de sécurité par exemple (arrêter une cuisinière laissée trop longtemps allumée, couper l'arrivée d'eau du bain

qui risque de déborder, *etc.*), alors l'interaction se fera entre machines. C'est le cas des échanges machine-homme que l'on discute ici.

Dans la continuité des interfaces saisissables, certains chercheurs se sont interrogés sur la possibilité d'utiliser le monde physique comme support pour la diffusion d'information digitale. L'idée est de rendre tangible cette information, et dans des modalités ne se limitant pas au seul paradigme visuel, qui prédomine aujourd'hui avec l'écran et les interfaces graphiques utilisateurs. L'expérience interactive de l'utilisateur en serait renforcé, lui dont les capacités de traitement de l'information sont limitées, tout particulièrement si la même modalité d'entrée est utilisée pour l'ensemble de l'information [128]. Ces travaux ont mis en avant le concept d'*interfaces tangibles*, à la fois interfaces saisissables et support pour une représentation tangible de l'information [150]. Au final, sans pour autant contredire le paradigme ubiquitaire, il n'est pas réellement ici question de disparition dans l'environnement de l'ordinateur mais plutôt de son incorporation tangible dans les objets du quotidien, faisant de ces derniers une catégorie d'objets communicants [128]. La notion de fonctionnalité reste, mais l'environnement de la personne évolue pour être dorénavant constitué de médias ambiants [74]. Le rapport de la personne avec le monde de l'information est complètement redéfini : la diffusion va se faire en arrière-plan, la personne étant libre de se concentrer sur ce que bon lui semble, l'information située à la périphérie de son attention pouvant prendre le dessus à tout moment si nécessaire, conformément au fonctionnement attentionnel humain.

De ces démarches novatrices, on retiendra la notion de médias ambiants, que l'on s'efforcera d'appliquer. Sur la forme, le plus juste serait de parler d'environnement augmenté (par exemple, [142]). Par cette notion, on désignera un environnement traditionnel dont certains des éléments physiques ont été équipés d'un dispositif technologique qui en augmente la capacité d'interaction. On souhaite ainsi pouvoir placer l'information d'assistance au coeur de l'activité et améliorer la compréhension et donc l'efficacité des *actes d'assistance* sous-jacents. Dans la cuisine, les endroits clés (placards, tiroirs, *etc.*) pourraient être équipés d'un dispositif local de "mise en valeur", comme un éclairage ou un dispositif sonore. Par défaut, ces éléments mobiliers, dont le rôle est pourtant significatif dans la réalisation des AVQ locales, ne sont doués d'aucune capacité d'interaction. C'est d'ailleurs l'importance de ce rôle qui devrait déterminer l'équipement. L'ensemble pourrait cohabiter avec des équipements d'interaction plus traditionnels, comme les écrans tactiles. Ces derniers, qui matériellement ne joue de rôle particulier dans la réalisation, pourraient offrir une vue plus macro de l'activité à réaliser. Petit à petit, c'est l'ensemble

de l'environnement qui devient support de communication entre la machine et l'homme.

5.2.1 IHM et communication

Pour aller au delà de l'IHM traditionnelle, il est fait grande référence à la notion de communication entre humains. C'est notamment le cas avec la prise en compte des interactions implicites. En poussant cette réflexion à l'extrême, certains auteurs penchent pour la mise en oeuvre d'*interactions sociales* entre l'homme et la machine. L'environnement devient alors *espace de communication*, chaque constituant de cet espace, homme ou machine, devenant possible émetteur et récepteur de messages de communication [130]. La modalité de prédilection est alors la voix, avec du côté machine des outils de reconnaissance de la parole et de synthèse vocale, le mimétisme étant poussé jusqu'à donner à la machine un visage sous la forme d'un avatar exprimant des émotions, pour que l'utilisateur trouve en la machine un "véritable" interlocuteur [93].

Dans le cadre des orthèses cognitives, la notion de communication, quelle que soit la modalité d'interaction choisie, ressort pleinement du processus d'assistance. Imaginons par exemple qu'au cours de la réalisation d'une recette de cuisine la personne prenne dans un placard de l'eau de Javel en lieu et place d'une bouteille d'huile. La logique d'assistance du système pourrait alors être comme suit :

1. étant donné l'erreur commise, l'orthèse "rédige" un message d'assistance adapté, invitant sur le fond à faire l'échange des bouteilles, qu'elle destine à la personne assistée ;
2. le message rédigé, l'orthèse va chercher un moyen de transmettre ce message, en fonction de ses capacités technologiques mais aussi des préférences de la personne, de son emplacement, *etc.* ;
3. si l'orthèse s'y prend judicieusement, la personne devrait recevoir le message, et le comprendre ;
4. une fois le message reçu et compris, la personne devrait agir en conséquence. On peut voir cela comme un accusé de réception, se traduisant par le rangement de l'eau de javel et la prise de l'huile.

Dans cette logique d'assistance, on retrouve différentes caractéristiques bien connues des modèles théoriques de communication, issus du monde de la linguistique et de la sociologie (voir par exemple, [27]). L'étude de ces modèles et de leurs différences dépasse

largement le cadre de cette thèse. Néanmoins, on peut reformuler ce processus de communication visant à délivrer un *acte d'assistance* en s'inspirant des caractéristiques principales de ces modèles :

- **Le message** La notion de *message* est au coeur de tout processus de communication. Dans le cas présent, deux notions sont à considérer :
  - **Le type du message** Un message est délivré vis-à-vis d'une situation, d'un contexte. Dans le cadre de l'assistance cognitive, ces situations vont correspondre aux erreurs commises par la personne. L'une des idées développées dans ce travail est d'arriver à formaliser ces erreurs, les problèmes de planification, attention, mémoire, initiation en restant les grandes catégories. A partir de là, on peut envisager associer des *types de messages* à ces catégories. Si par exemple la personne oublie de réaliser une activité, un message de type "rappel d'activité" semble approprié. La finalité de cette approche est de faire le lien entre le type de message, concept abstrait, et la délivrance du message par un dispositif de type effecteur dans l'environnement. De par ses spécificités techniques, tout effecteur n'aura pas la capacité de diffuser tous les types de messages. On parlera alors à nouveau de service d'IHM. Enfin, pour un même type de message, plusieurs effecteurs pourront assurer de manière différente la diffusion de l'information ;
  - **Le contenu du message** Le *contenu* correspond à l'information que l'orthèse souhaite communiquer à la personne. En fonction du type de message, la nature de l'information pourra varier, de simple donnée à structure complexe, comme une procédure de réalisation d'une activité.
- **L'émetteur** L'*émetteur* est bien entendu l'entité à l'origine du message. En l'occurrence, il devrait s'agir d'une fonctionnalité de l'application ;
- **Récepteur** Le *récepteur* est le destinataire du message. Dans le cas présent, le client ou l'un de ses tiers devrait être concerné ;
- **Temporalité** La *temporalité* fait référence au temps de persistance du message, autrement dit au temps pendant lequel l'information sera présentée au récepteur. Divers scénario peuvent être envisagés : durée fixe, conditionnée à la confirmation de la réception du message, liée au contenu du message, *etc.* ;
- **Localisation** Un message peut être *localisé*, autrement dit sa diffusion sera concentrée

à un endroit précis de l'environnement, soit *alocalisée*, ce qui signifie que l'information devra être, dans l'absolu, accessible en tout endroit de l'environnement ;

➤ **Rétroaction** Une *rétroaction* sera vue ici comme la confirmation de la réception du message, par le récepteur.

Requête d'interaction homme-machine

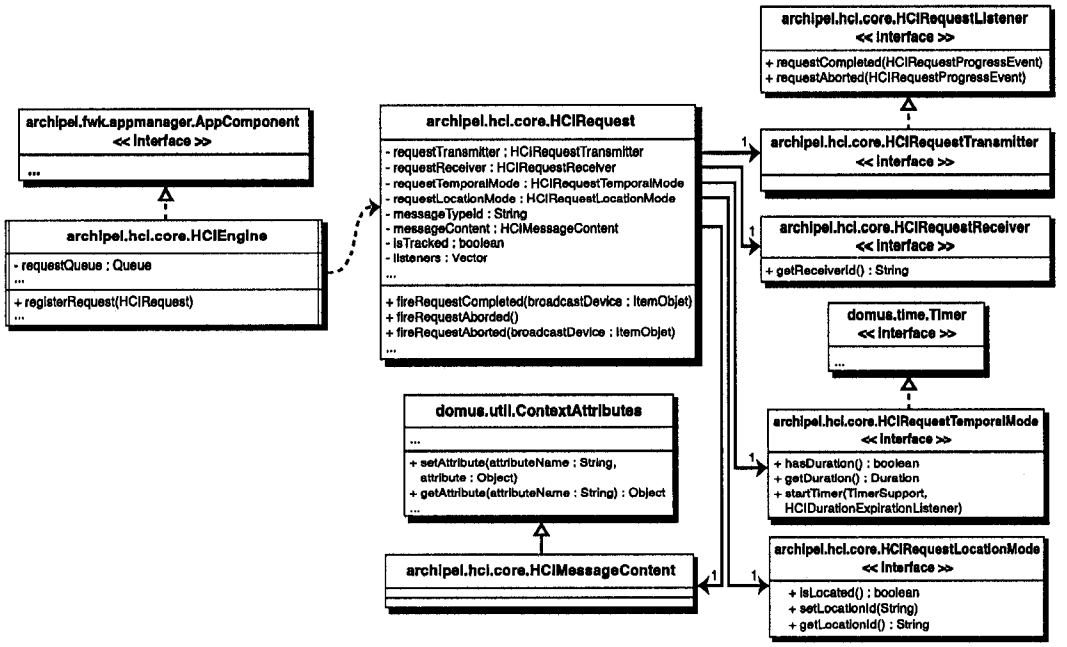

FIGURE 5.2 – Diagramme des classes de modélisation des requêtes d'IHM (vue partielle)

De cet énoncé des caractéristiques est issu le diagramme de classes présenté partiellement à la figure 5.2. L'élément fédérateur de ces classes est la *requête d'IHM* qui, formulée par un objet émetteur, par exemple une fonctionnalité de monitorage d'activité, est soumise pour traitement au moteur d'IHM. Le rôle principal du moteur est de résoudre les requêtes formulées, c'est-à-dire (1) de trouver l'ensemble des dispositifs matériels permettant la diffusion du message d'assistance et (2) d'invoquer cette diffusion. Ces deux questions sont à nouveau abordées sous l'angle de l'utilisation des connaissances représentées, grâce à archipel.lang. La première concerne la description des effecteurs, des services d'IHM qu'ils fournissent, et des types de message. L'ensemble utilise une mécanique de

représentation proche de celle utilisée pour les capteurs. Trouver l'ensemble des dispositifs revient à connaître l'ensemble des instances de l'ItemObjet représentant le service associé au type de message.

Une fois l'ensemble des effecteurs adéquats obtenu et si la requête est localisée, le moteur effectue une sélection selon le critère de localisation. L'emplacement d'un effecteur ou de tout autre entité contextuelle est décrit comme une simple propriété dans le modèle de connaissance, prenant pour valeur un item issu de la représentation du contexte spatial. Ce dernier est décrit selon un point de vue *ensembliste* [55], sous la forme d'inclusions des espaces les uns dans les autres. Concrètement, chaque item modélisant un espace nommé de l'environnement, comme une pièce, dispose de la liste des espaces qu'il inclut (espace plan de travail, espace cuisinière, espace micro-onde, *etc.*). Parcourir ces inclusions s'apparente alors à celui d'un arbre. A noter que l'intérêt de ce découpage réside dans sa capacité à servir les interactions homme-machines. Il est donc parfaitement arbitraire. Ainsi, cette représentation très simple est utilisée par le moteur d'IHM, pour raisonner sur la capacité spatiale d'un effecteur à réaliser l'interaction. Si le dispositif se trouve dans l'"espace cuisinière" et que le message doit être diffusé dans la cuisine, alors l'effecteur pourra être retenu, de ce point de vue.

Diffusion du message

L'invocation de la diffusion du message repose sur la mise en oeuvre du patron de conception proxy[9] [57]. Le principe est le suivant : l'ItemObject modélisant l'effecteur dispose d'une propriété qui sert de mandataire ou disons d'interface avec l'effecteur réel. Cette propriété, un objet Java que l'on appellera *proxy*, sera typiquement issue d'une librairie de communication dédiée au type d'effecteur, ou servira d'adaptateur pour cette librairie. Une autre situation intéressante concerne les effecteurs de type écran, où une interface web peut être utilisée comme support de présentation de l'information. C'est alors la servlet[10] dédiée à l'interface web qui pourra être vue comme le *proxy* de l'écran.

Pour pouvoir remplir sa mission, l'objet *proxy* doit comprendre un message spécifique dont le sélecteur est précisé au niveau du service d'IHM, l'argument n'étant autre que la requête d'IHM. La méthode correspondante aura pour tâche de fournir le service conformément aux capacités d'interaction de l'effecteur, la notion de service allant au

[9]Terme anglo-saxon utilisé de manière courante en informatique. "Mandataire" serait une traduction française appropriée.

[10]Objet Java utilisé par un serveur web pour gérer les requêtes émanant d'un client de navigation.

delà des considérations matérielles d'interaction. Du point de vue du moteur d'IHM, la gestion de ce message est transparente : le moteur ne fait que travailler avec l'ItemObject représentant l'effecteur. C'est en effet à ce niveau que la gestion du message est adaptée, dans l'esprit du patron de conception *proxy* : le comportement de l'ItemObject va assurer, grâce au MOP d'interprétation des messages, la manipulation de ses propres ressources comportementales et la découverte dynamique - par introspection - des capacités de ses propriétés. Si la propriété *proxy* ne dispose pas de la signature adéquate, alors l'ItemObject ne pourra que manifester sa non-compréhension du message. A noter pour terminer qu'une mécanique événementielle assure à l'émetteur la connaissance de l'état de diffusion de sa requête.

Au final, cette mécanique d'interaction illustre le principe de *séparation des préoccupations.* Pour l'objet émetteur, le travail se limite à la génération d'une représentation abstraite de l'interaction, sous la forme d'une requête d'IHM. Pour le moteur d'IHM, il s'agit de raisonner sur les effecteurs décrits puis d'invoquer la diffusion par envoi de message. Du côté effecteur, c'est l'objet *proxy* qui est en charge de la concrétisation de cette interaction. Ainsi, l'émetteur n'a pas à se soucier du média de transmission, ce qui permet une modularité côté effecteurs de même nature que pour les capteurs.

5.2.2 Boucler la boucle

Archipel.hci ne présente pas de modèle global d'IHM. En effet, comme cela vient d'être illustré, d'un côté existe la gestion des interactions homme vers machine et de l'autre celle de la diffusion des messages d'assistance (machine vers homme). Néanmoins, avec la prise en compte des *rétroactions* du client, un point de jonction est établi entre ces facettes.

La rétroaction, qui est une IHM, est vue comme la confirmation de la *réception* du message. Or même si celle-ci n'implique pas forcément la *compréhension* de ce message, elle en constitue la première étape. Si l'interaction de rétroaction est explicite, ce qui sous-entend que le système ait offert à la personne le moyen de faire part de sa bonne réception du message, la situation est simple. Si par contre aucune possibilité explicite n'est offerte, comment s'assurer de la réception, et possiblement de la compréhension du message ? En fait le problème est plus ciblé dans Archipel : les messages sont envoyés suite à la survenance d'une erreur. Dès lors, on admettra que la correction de cette erreur sous-entend la bonne réception et *a fortiori* compréhension du message. C'est un principe

un peu grossier, mais qui semble le plus sage à ce stade de la recherche.

Le besoin de la prise en compte de la rétroaction se fait particulièrement sentir pour gérer la temporalité des communications. Le cas typique est la mise en évidence d'un emplacement par un dispositif d'éclairage, comme un tiroir qui contiendrait les ustensiles que la personne ne semble pas trouver. La question est combien de temps doit-on laisser allumer l'éclairage? La réponse la plus sage est *tant que la personne n'a pas ouvert le tiroir*. L'événement d'ouverture constituera alors la rétroaction attendue par l'effecteur pour s'éteindre. Du point de vue du traitement de l'interaction, la rétroaction implicite sera "calculée" au moment de la résolution de la requête, si le type du message le suggère. Ce calcul correspond à l'établissement de la relation spatiale la plus précise entre l'emplacement de l'émission et celui de l'interaction, soit dans le présent exemple l'événement d'ouverture du placard pour l'utilisation de l'effecteur de "mise en évidence" du même placard.

5.3 Quelques réflexions conclusives

Grâce à son utilisation d'archipel.lang, archipel.hci offre une solution modulaire mais aussi intégrée pour la gestion des interactions homme-machine. Les déploiements réalisés, présentés au chapitre 8, montrent l'intérêt de cette solution pour l'assistance cognitive. L'ensemble de l'environnement ou tout du moins de ses artefacts devenant possiblement support d'interaction, il serait pertinent d'évaluer les capacités du *framework* dans un contexte général de gestion de contexte, tout particulièrement en ce qui concerne la mise en oeuvre des interactions implicites.

Toutefois, dans un contexte d'informatique ubiquitaire, l'IHM fait état de nombreuses autres problématiques. On pense par exemple à la localisation et l'identification du ou des occupants [55,129], ou encore à la distribution et à la migration des interfaces [39]. De même, on peut souhaiter relativiser la survenance d'événements atomiques par le biais de méthodes probabilistes, afin de prendre en compte le degré de fiabilité des capteurs, ou effectuer une intégration de plus haut niveau des interactions implicites atomiques, afin de fournir au niveau applicatif une vue "macro" des événements. A la question archipel.hci peut-il supporter toutes ces évolutions il est impossible de répondre en l'état des travaux. Néanmoins, on soulignera son développement sous forme de *framework* et les capacités d'évolution en découlant.

Mais au delà de ces considérations techniques, une problématique tout aussi complexe

si ce n'est plus réside dans l'essence même de ce travail : interagir de façon judicieuse avec les personnes présentant des troubles cognitifs. D'une manière générale, il n'a pas été fait mention dans ce chapitre de la prise en compte des préférences de l'usager pour l'interaction, ce problème dépassant d'ailleurs le cadre de l'assistance cognitive. Des travaux ont ainsi été réalisés dans ce sens pour les personnes avec troubles moteurs (voir par exemple [76]). Dans Archipel, une première solution simple résiderait dans la redéfinition de la stratégie de choix des effecteurs au niveau du moteur d'IHM, pour la prise en compte de critères explicites comme la non-capacité de lecture ou les problèmes d'audition. Du point de vue de la séparation des préoccupations, les ajustements de ce type ont leur place à ce niveau du traitement de l'interaction. Il s'agit en effet de réglages très concrets, qui doivent prendre place à la porte de la diffusion de l'information. Par contre, au niveau de la génération de la requête les choix les plus délicats demeurent : niveau d'abstraction du message, moment de la diffusion, suivi de la réalisation, réajustement du contenu, *etc.* Ces problématiques complexes seront en partie abordées dans les prochains chapitres, et leur mise en oeuvre discutée lors de la présentation des résultats.

Chapitre 6

archipel.monitoring

Si chaque facette du processus de sensibilité au contexte d'assistance cognitive joue un rôle significatif dans la production des actes d'assistance, suivant une certaine boucle de contrôle déjà présentée, la compréhension du CAC en est clairement l'élément central. Etant donnés les événements contextuels perçus, l'étape de compréhension doit en effet permettre la détection des erreurs commises par la personne et leur qualification, donnant au système sa capacité de raisonnement la plus significative et ouvrant ainsi la voie à la mise en oeuvre d'une stratégie d'assistance adaptée.

Pour réaliser une telle analyse, il faut doter le système d'une mécanique permettant d'interpréter la survenance des événements contextuels. Le cadre de cette survenance étant celui de la réalisation d'AVQ et l'objectif global l'effectivité de cette réalisation, la référence pour l'interprétation ne peut être que l'ensemble des bonnes façons de réaliser la ou les activités en question, c'est-à-dire celles pour lesquelles un besoin a été exprimé. Dès lors, on doit s'interroger sur la nature de cette analyse : le problème est-il de reconnaître l'activité ou d'en suivre la réalisation ? Cette question fera l'objet de la première section de ce chapitre, ouvrant la voie au principe de monitorage d'activités. La représentation de ces dernières sera ensuite abordée, puis le lien entre contraintes de réalisation et diagnostic établi, dans l'esprit des troubles cognitifs introduits au premier chapitre. Enfin, on s'intéressera à deux techniques de monitorage sous la forme d'études de cas : la première concernera les activités que l'on qualifiera de *domotiques*, dont le principe de suivi est des plus simple. La seconde portera sur les activités domiciliaires complexes, avec une mise en oeuvre basée sur le système *EpiTalk* [114], issu du domaine des systèmes conseillers. Le tout permettra d'illustrer le fonctionnement du sous-*framework* archipel.monitoring, dédié à la compréhension du contexte d'assistance cognitive.

6.1 Monitorage versus reconnaissance

On peut envisager l'analyse des événements contextuels selon plusieurs points de vue, en fonction des objectifs de l'application. Le *monitorage* et la *reconnaissance* d'activité sont deux approches possibles qui s'inscrivent dans des démarches de nature clairement différente. Pour autant, il semble exister dans certains travaux une confusion autour de ces notions ou peut être tout simplement un abus de langage qui n'a pas à être. L'objectif de cette section est de poser les différences entre monitorage et reconnaissance et de réfléchir sur la nature de l'analyse des événements pour une application d'assistance à la réalisation d'activités.

La reconnaissance d'activité s'apparente au problème générique de reconnaissance de plan, discipline de l'intelligence artificielle bien établie. D'après l'ouvrage *Automated Planning : theory and practice* [58], le problème de reconnaissance de plan se présente de la façon suivante : dans un monde cohabitent deux agents, un *acteur* et un *observateur*. Le premier effectue une séquence d'actions, tandis que l'autre essaie de comprendre le ou les plans suivis par l'acteur dans le but d'inférer son ou ses buts courants. Trois types d'approches peuvent alors intervenir :

- l'acteur coopère avec l'observateur et tente de lui communiquer clairement ses intentions, c'est le problème de reconnaissance de plan intentionnel ;
- l'acteur tente de cacher ses intentions à l'observateur. Cette problématique est plus particulièrement visée par les applications de surveillance ou à usage militaire ;
- l'acteur est neutre, car sensé ignorer la présence de l'observateur. On parle alors de *keyhole plan recognition problem*[1], l'observation se faisant à l'insu de l'acteur. C'est l'approche la plus populaire.

On ne rentrera pas dans les considérations techniques de la reconnaissance de plan, dont un portrait détaillé peut être trouvé dans la thèse de B. Bouchard [24]. On se posera par contre la question suivante : soit l'orthèse cognitive l'observateur et le client l'acteur, l'objectif de l'orthèse est-il réellement d'inférer le ou les buts courants du client, autrement dit la ou les activités que la personne tente de réaliser ? A cette question on répondra par la négative : l'objectif de l'orthèse est d'assurer qu'étant donnés les besoins en terme d'activités exprimés par la personne (ou son entourage), il y a bien effectivité de la réalisation. Ce n'est donc définitivement pas de la reconnaissance de plan, mais du suivi de la réalisation d'activités dont il est question ici. Les différences les plus significatives

[1] *Keyhole* = trou de serrure.

sont la connaissance *a priori* des buts (i.e. des activités à réaliser) de la personne, dont découle un nouvel objectif pour l'orthèse à ce niveau du raisonnement : déceler parmi les actions de la personnes celles qui ne permettent pas l'effectivité de la réalisation. A partir de là, la personne peut en effet coopérer avec l'orthèse, tenter de cacher ses intentions ou rester neutre. Mais dans un premier temps, on considérera que le client est conscient de la présence de l'orthèse et l'accepte en tant que telle, autrement-dit qu'il n'agit pas de manière à l'induire consciemment en erreur. On peut supposer que si tel étais le cas, alors une telle orthèse ne serait pas un dispositif d'assistance adapté à la personne. L'explication pourrait se trouver du côté humain ou du côté machine, avec un problème d'utilisabilité de l'orthèse.

Pour cette problématique, on a déjà proposé le terme de *monitorage d'activité* ([122] et [123] - version étendue du premier). Monitorage est la recommandation officielle pour *monitoring*, anglicisme courant dans la langue française. D'après le dictionnaire Petit Robert 2007, le *monitoring* est en médecine une "technique de surveillance médicale électronique, au moyen de capteurs enregistrant différents paramètres et de systèmes d'alarme se déclenchant en cas d'écart des valeurs physiologiques", et dans le domaine des technologies une "technique de surveillance de systèmes complexes, assistée par ordinateur". Bien qu'on ne puisse parler de "surveillance" pour la présente problématique, le principe reste sur le fond pertinent. Ainsi, dans le domaine des orthèses cognitives, on définira par "monitorage" une technique de compréhension du contexte d'assistance cognitive permettant le suivi de la réalisation d'activités prédéfinies, se concrétisant par la détection et la qualification des erreurs commises.

A partir de là, il est intéressant de regarder ce que font réellement les orthèses présentes dans la littérature, pour la plupart déjà citées dans ce mémoire, mais que l'on présentera dans ce chapitre sous un angle nouveau. COACH [35] ne permet l'assistance, tout du moins dans sa version actuelle, que du lavage des mains. Le but de la personne étant parfaitement connu, il n'y a pas de reconnaissance mais bien un suivi de la réalisation. La référence pour ce monitorage est alors un plan optimal de réalisation de l'activité (une séquence d'actions), initialement calculé par le système. Du point de vue de la compréhension du CAC, chaque action correspond à un événement contextuel, obtenu de façon implicite grâce à une caméra qui détecte les mouvements des mains et un débitmètre pour l'utilisation la robinetterie. Comme traditionnellement en planification, on peut voir une action comme une transition d'un état vers un autre, un état correspondant à une situation contextuelle. Ainsi, ce que le système calcule, c'est l'ensemble des états

ou situations dans lesquels le client peut se trouver, d'après les événements ou actions détectables. Dès lors, du point de vue du monitorage, on ne peut pas réellement dire que COACH s'efforce de détecter les erreurs et de les qualifier. Ce qu'il fait, c'est prendre acte de l'état dans lequel se trouve la personne, et à partir de là produire un acte d'assistance sensé inciter la personne à effectuer une action correspondant à une transition vers un état plus abouti, du point de vue de la réalisation de l'activité et du plan optimal. C'est pourquoi on reviendra sur cette approche dans le prochain chapitre, qui portera sur l'assistance.

Le second exemple est celui d'Autominder, un dispositif destiné à pallier les troubles mnésiques prospectifs chez les personnes âgées (Autominder pour *reminder*, rappel) [34, 101]. Autominder se place pour sa part sous l'angle de la reconnaissance d'activité. L'orthèse a une connaissance *a priori* de l'ensemble des buts que la personne doit réaliser sur la journée (besoins explicitement exprimés par la personne ou un intervenant), un ensemble de contraintes temporelles étant associées à chaque but (i.e. activité à réaliser). A partir de là, l'objectif d'Autominder est de reconnaître parmi les but exprimés quels sont ceux réalisés par la personne, et lorsqu'une activité manque à l'appel d'effectuer le rappel nécessaire. Les événements contextuels sont par contre limités, car le dispositif est monté sur un robot mobile sensé suivre la personne et ne disposant que de quelques capteurs. Une technique d'inférence probabiliste est alors utilisée pour effectuer la reconnaissance. Malheureusement, Autominder n'a pas été expérimenté auprès de la clientèle. Il aurait été intéressant de voir dans quelle mesure la reconnaissance d'activité telle que mise en oeuvre peut servir de support à l'assistance. Plus précisément, la reconnaissance doit permettre à l'orthèse de limiter ses rappels aux activités qui n'ont effectivement pas été réalisées. Cette thématique est toujours présente dans la nouvelle version d'Autominder qui est en cours d'élaboration, le support robotique ayant par contre été mis de côté au profit d'un déploiement plus ubiquitaire [155].

Pour finaliser cette discussion, l'exemple d'Opportunity Knocks est des plus pertinents, car il combine reconnaissance et monitorage [151]. Opportunity Knocks s'attaque à l'aide au déplacement. Grâce à sa technique d'apprentissage non supervisé, il constitue dans un premier temps un catalogue des chemins habituels de la personne. Lorsqu'il dispose de cette connaissance, Opportunity Knocks, qui est installé sur un téléphone portable couplé à un récepteur GPS, est à même de proposer à la personne des opportunités de déplacements, en faisant le lien entre sa position actuelle et ses habitudes de déplacement. Il s'agit alors clairement de reconnaissance d'activité, puisque le système

infère l'ensemble des buts possibles de la personne. A partir de là, la personne peut sélectionner parmi ces opportunités celle pour laquelle elle veut être assistée. On rentre alors dans une deuxième phase, qui consiste au monitorage de l'activité de déplacement choisie (le but est maintenant explicitement connu). Enfin, si le parcours réalisé n'est pas conforme à l'objectif, Opportunity Knocks vérifie l'éligibilité de ce mauvais parcours comme autre habitude de déplacement. La reconnaissance d'activité est alors utilisée pour qualifier l'erreur commise. Malheureusement, aucune expérimentation avec la clientèle n'a été effectuée avec Opportunity Knocks.

En conclusion, l'assistance à la réalisation d'activités s'inscrit clairement dans une thématique de monitorage. La question est alors de savoir ce qui permet de déterminer qu'à un moment donné, le système se doit d'aider la personne à réaliser telle activité[2]. La première solution réside dans une demande explicite de la personne. Celle-ci pourrait être guidée par un "catalogue" d'activités, éventuellement contextualisé. C'est le cas d'Opportunity Knocks, où les activités de déplacement sont proposées en fonction de l'emplacement courant. Cela pourrait être appliqué très simplement en intérieur, si les activités disposent d'une caractéristique ou précondition spatiale (par exemple, une activité de recette ne peut être réalisée que dans la cuisine). La caractéristique pourrait tout aussi bien être temporelle, sous la forme d'un emploi du temps. Il pourrait y avoir alors combinaison d'assistance, soit un rappel d'activité puis une proposition d'aide à la réalisation, éventuellement automatique pour certains profils d'assistance. On retombe alors sur le problème du non-rappel d'activités déjà réalisées. Dans la même lignée, l'objectif pourrait être de s'assurer qu'une activité n'est pas réalisée en dehors d'une plage horaire donnée, comme une prise de médicaments. Dans ce cas précis, le but est connu mais le problème inversé, puisqu'il faut s'assurer de sa non-réalisation. Cela revient en quelque sorte à monitorer à longueur de journée l'activité en question.

6.2 Monitorage, détection et qualification des erreurs

Le modèle servant de référence pour le monitorage se doit de témoigner de la ou des bonnes façons de réaliser l'activité. Cette section présente le modèle utilisé par Archipel, qui supporte la description des contraintes de réalisation dont la violation détermine les erreurs et leur qualification.

[2]L'opportunité d'aider la personne à réaliser plusieurs activités simultanées a déjà été discutée au chapitre 3.

6.2.1 Représentation et interprétation des activités

Actuellement, la représentation des activités est basée sur deux modèles complémentaires. Le premier est *ad hoc* et supporte la représentation de la structure des activités et de leurs contraintes. Le second permet la spécification de la sémantique des activités, et utilise archipel.lang.

La représentation de la structure des activités est issue du travail de maîtrise de l'auteur [13,15]. Le modèle développé offre la souplesse nécessaire à une description générique des activités. Sa représentation hiérarchique est proche des modèles utilisés en neuropsychologie [37] et en planification par réseau de tâches hiérarchiques [58]. A chaque tâche est associée un ensemble de méthodes, qui correspondent à autant de façons de réaliser la tâche. Une méthode est logiquement un ensemble de sous-tâches, donnant ainsi naissance à la structure hiérarchique. La sémantique des tâches racines est celle des AVQ, les feuilles étant les méthodes des tâches terminales, qui représentent des actions concrètes de la personne dont l'exécution doit être perceptible dans l'environnement. Les niveaux intermédiaires permettent une décomposition intelligible de l'activité. Le modèle permet de plus la spécification des contraintes de réalisations. Celles-ci peuvent être temporelles (durée, plage horaire de réalisation), ou comportementales (contraintes d'ordonnancement, de répétition, d'optionnalité, *etc.*). Enfin, à chaque tâche ou méthode peuvent être associés préconditions et effets. Ceux-ci concernent l'environnement de réalisation, et correspondent à la valeur *a priori* (précondition) ou *a posteriori* (effet) d'une propriété d'un item. Autrement dit, si la réalisation d'une activité nécessite la présence de la personne dans la cuisine, alors une précondition portant sur la propriété "location" (ou autre propriété décrivant l'emplacement) de l'item représentant le client devra être associée à la tâche racine du modèle. Le problème de la représentation de la sémantique des tâches *via* archipel.lang sera pour sa part discuté ultérieurement, celle-ci étant essentiellement utilisée pour la génération de l'assistance.

Les actions associées aux méthodes des tâches terminales correspondent bien entendu à des événements d'interaction implicite ou explicite. Comme cela a été précisé au chapitre 5, la solution retenue, si l'environnement n'est que partiellement observable pour une tâche donnée, est de demander explicitement à l'utilisateur de confirmer cette réalisation. Puisqu'il s'agit d'une solution par défaut, les interactions explicites n'ont pas à être précisée au niveau de la représentation.

Du côté des interactions implicites, deux possibilités existent : soit la survenance de cet événement d'interaction implique de façon catégorique la réalisation de la tâche ter-

minale (par exemple, événement d'ouverture de la porte du réfrigérateur pour une tâche dont le but est précisément l'ouverture de cette porte), soit la survenance n'est qu'un indice de cette réalisation (événement d'ouverture de la porte du réfrigérateur pour une tâche dont le but est de prendre dans ce réfrigérateur la boîte d'oeufs). Dans le premier cas, l'événement sera qualifié de *discriminant*. Dans le second cas, l'événement non discriminant sera vu comme une condition nécessaire (ouvrir le réfrigérateur pour y prendre les oeufs, qui ne disposent pas de leur propre système de captation) mais non suffisante (une fois le réfrigérateur ouvert, tout objet s'y trouvant peut avoir été pris à la place des oeufs) pour la réalisation de la tâche terminale. Puisque l'environnement n'est à nouveau que partiellement observable, il sera demandé à la personne une confirmation explicite de la réalisation. Néanmoins, on nuancera cette confirmation, qui peut parfaitement être sans fondement, si l'indice que représente la condition nécessaire mais non discriminante n'a pas été validé. On augmente ainsi la capacité de monitorage du système.

Du point de vue de la mise en oeuvre, l'ensemble des connaissances structurelles est décrit en XML et manipulé à l'exécution selon un modèle objet Java. Sans aller plus en avant dans ces remarques techniques, on précisera l'existence d'un éditeur graphique dont une copie d'écran est présentée à la figure 6.1. Y figure une vue partielle de la structure d'une activité de recette, les icônes "T" et "M" représentant respectivement les tâches et méthodes. On notera dans la partie gauche le catalogue des tâches et des méthodes, les premières étant organisées selon leur sémantique. Tâches et méthodes associées sont clairement vues comme des briques de connaissance réutilisables, que l'on peut composer et éditer graphiquement. On remarquera d'ailleurs dans la partie basse la feuille de propriétés de la tâche sélectionnée, donnant notamment accès à sa durée, ses préconditions et ses effets.

6.2.2 Contraintes et diagnostic

Traditionnellement, dans les applications ubiquitaires, le contexte caractérise la situation dont dépendent les actions à prendre [46]. Avec le monitorage des activités, Archipel introduit une étape supplémentaire : la qualification des erreurs, ou phase de diagnostic. Autrement dit, le contexte caractérise la situation qui, si elle témoigne d'une erreur dans la réalisation de l'activité, donne lieu à la génération d'un diagnostic. Et c'est du diagnostic dont vont dépendre les actions, i.e. les actes d'assistance.

Cet ajout ne révolutionne en rien le principe de base. On peut d'ailleurs simplement

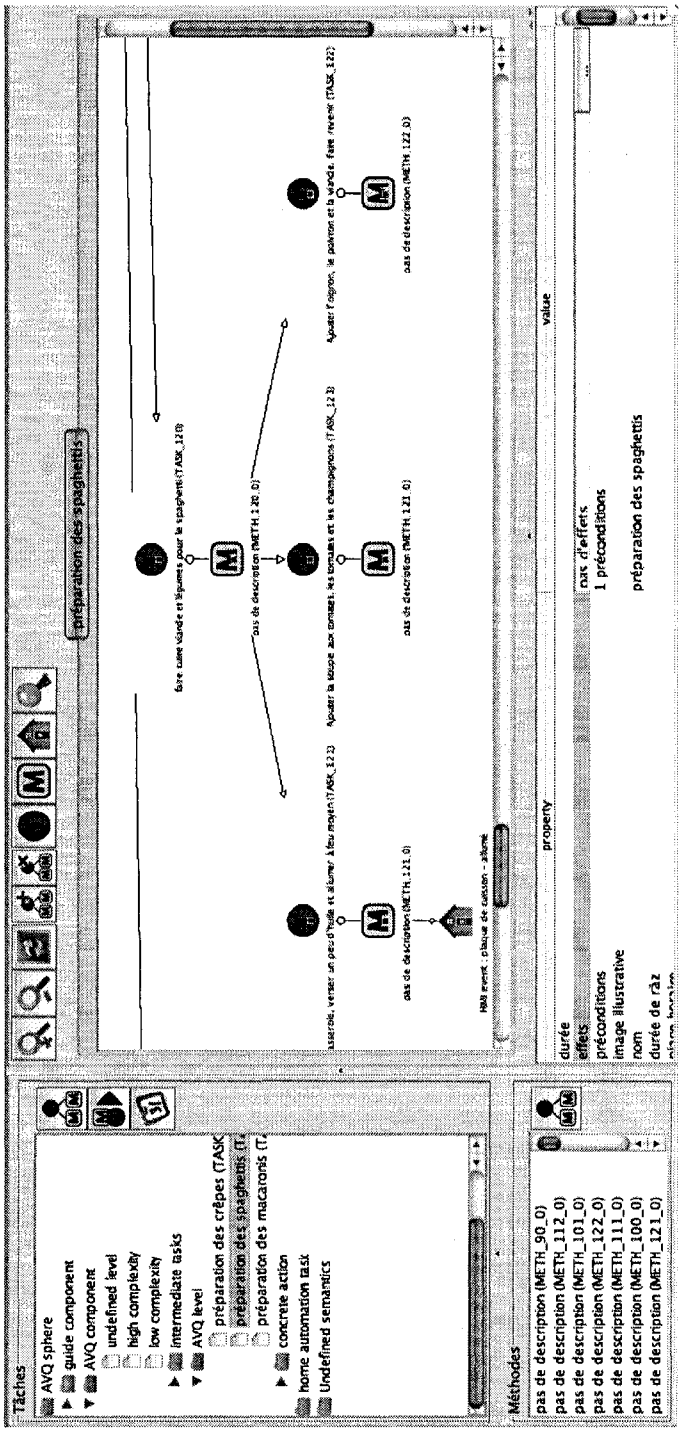

FIGURE 6.1 – Composition et édition graphique des activités

le voir comme un niveau d'interprétation/abstraction supplémentaire. L'objectif est en effet de systématiser les erreurs de réalisation et d'offrir ainsi au système un vocabulaire autour duquel il pourra construire ses stratégies d'assistance, et qui permettra aussi côté utilisateur l'écriture de profils d'assistance. Ce sont bien entendu les troubles cognitifs présentés au chapitre 1 qui serviront alors de référence.

Des liens ont donc été créés entre erreurs de réalisation telles que comprises côté informatique et troubles, ou plus précisément entre violation des contraintes de réalisation et troubles, pour permettre la génération "algorithmique" de diagnostics. On conviendra toutefois qu'à ce stade de la recherche, cette approche ne constitue absolument pas la transcription informatique d'une expertise médicale. En voici la description, présentée dans [18] :

Diagnostic de trouble de l'initiation Est posé lorsqu'il y a absence d'événements contextuels pendant une durée significative alors que la personne doit débuter une activité. La présence d'une activité à débuter découle de la description des besoins de la personne. La notion de durée significative dépend de chaque personne et devrait être précisée par son profil. Enfin, l'absence d'événements s'applique aux événements en lien avec l'activité ;

Diagnostic de trouble de la planification Ce type de diagnostic est clairement lié à la violation de contraintes d'ordonnancement. Trois situations peuvent se produire :

1. l'événement est en lien avec l'activité en cours, mais celui-ci n'aurait pas dû intervenir à ce moment de la réalisation. Autrement dit, un certain nombre d'étapes (i.e. sous-tâches) sont manquantes pour satisfaire la (ou les) séquence(s) possible(s) de réalisation ;
2. l'événement n'est pas en lien avec l'activité, mais est conforme à la contrainte de localisation de cette dernière. La personne semble donc éprouver de la difficulté à formaliser la prochaine étape à réaliser ;
3. l'activité suit son cours, mais le temps moyen de réalisation de l'étape courante est dépassé. A nouveau la personne semble éprouver de la difficulté à formaliser la suite de la réalisation.

Diagnostic de trouble de l'attention Il y a trouble de l'attention lorsque l'événement n'est pas en relation avec l'activité courante et prend place dans un lieu autre que celui de réalisation de l'activité ;

Diagnostic de trouble mnésique La qualification des troubles mnésiques semble plus délicate. Ceux-ci peuvent en effet se confondre, du point de vue informatique, dans les problématiques précédentes. Par exemple, une mémoire à court terme déficiente, qui amènerait la personne à réaliser de nouveau une même étape de réalisation, serait comprise comme un problème de planification. Quoi qu'il en soit, une approche différente est utilisée pour ce type de qualification : si l'on suppose que la personne a conscience de ses troubles mnésiques, elle pourrait être amenée à demander assistance au système, qui s'efforcera de lui rappeler l'information oubliée. Dès lors, on se basera sur la nature des demandes pour qualifier un tel trouble.

En conclusion, on comprend clairement que cette démarche de diagnostic vise à donner à l'orthèse des outils pour essayer de comprendre les difficultés de son client. Bien entendu, cette approche reste à l'heure actuelle très grossière. On la considérera néanmoins comme une première exploration de la question.

6.3 Principes de monitorage : deux études de cas

Le *framework* archipel.monitoring a pour but le suivi de la réalisation d'activités présélectionnées, la détection des erreurs et leur qualification, selon l'approche précédemment décrite. Dans cette section sont présentées les mises en oeuvre effectuées pour atteindre ces objectifs, illustrées par deux études de cas.

6.3.1 Principes généraux du *framework* archipel.monitoring

Le *framework* offre un cadre abstrait pour la mise en oeuvre du monitorage, dont la figure 6.2 présente les principaux composants. Toute classe définie par l'utilisateur destinée à servir de support pour le monitorage doit implémenter l'interface MonitoringUnit, faisant de toute instance de cette classe une unité de monitorage pour une activité donnée. La gestion du cycle de vie de l'unité est confiée à un MonitoringUnitManager (gestionnaire), qui se comporte comme un composant d'application au sens du *framework* archipel.fwk.appmanager (voir annexe E). Lorsqu'une activité doit être monitorée, celle-ci est soumise au gestionnaire. Celui-ci fait alors appel à une fabrique d'unité de monitorage (dans le sens du patron de conception *factory* [57]) pour créer l'unité adéquate. La fabrique utilise la représentation sémantique des tâches pour connaître le "créateur" d'unité à invoquer (sous-classe concrète de MonitoringUnitCreator). Dans la mise en oeuvre

actuelle, deux types de créateurs et donc d'unités de monitorage ont été développés, pour les activités domotiques et les activités domiciliaires complexes, soit les deux cas étudiés dans cette section.

Une unité de monitorage possède un *état de monitorage*, un *état d'achèvement*, ainsi qu'un *état temporel*. L'état de monitorage correspond au cycle de vie de l'unité, tandis que l'état d'achèvement décrit le degré de réalisation de la tâche monitorée. Lorsque l'état de monitorage passe à "actif", l'unité devrait rendre le monitorage effectif, tout particulièrement en demandant à être notifiée des événements contextuels concernant l'activité monitorée. Un autre état de monitorage important est "arrière plan", cet état signifiant que le client a demandé au système de ne plus l'assister dans sa réalisation. Du côté achèvement, un état intéressant est le "considéré comme complétée", qui s'applique à toute activité dont certaines ou l'ensemble des sous-tâches *optionnelles* n'ont pas été réalisées (ce qui ne doit pas être considéré comme une erreur). Enfin, les changements d'état temporel témoignent de la violation des contraintes temporelles de l'activité.

Une unité de monitorage est amenée à poser des diagnostics, qui intéresseront la partie de l'application chargée de gérer l'assistance. La communication des diagnostics, ainsi que des changements d'état et de la création d'unités de monitorage utilise la mécanique traditionnelle du patron observateur (interfaces MonitoringUnitListener et MonitoringUnitManagerListener) [57]. Un jeu d'événements appropriés est utilisé comme vecteur de communication.

6.3.2 Activités domotiques

Le premier cas de mise en oeuvre concrète du monitorage est relativement simple, et s'adresse aux *activités domotiques*. Par *domotiques*, on entend les activités de type action/réponse, c'est-à-dire répondant à un événement contextuel par une modification de l'environnement. Il pourrait par exemple s'agir de gérer l'éclairage lorsque la personne entre ou quitte une pièce. Ce cas ne rentrant pas dans le champ de la qualification des erreurs de réalisation, son intérêt est avant tout d'illustrer la capacité du *framework* à représenter et gérer différentes sortes d'activités.

Du côté modélisation, les activités domotiques sont décrites sous la forme d'un simple couple tâche/méthode. Un événement d'interaction est référencé côté méthode, la tâche possédant pour sa part un ensemble d'effets environnementaux. L'unité de monitorage créée se présente sous la forme d'un processus autonome dont le rôle est d'appliquer les

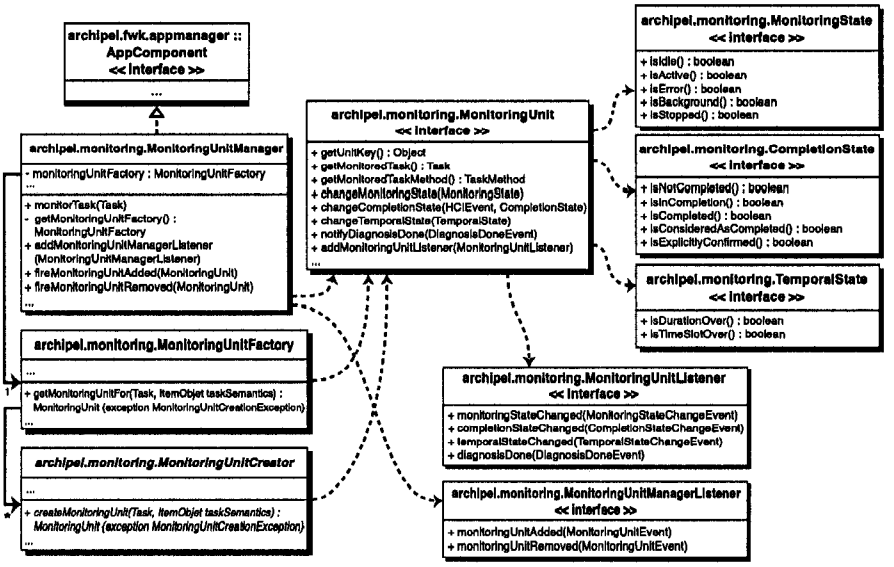

FIGURE 6.2 – Diagramme partiel des classes, *framework* archipel.monitoring

effets à chaque fois que l'événement est perçu, et ce tant que le monitorage est actif, sauf indication contraire côté sémantique de l'activité (l'application des effets pourrait n'être qu'unique, celle-ci entraînant le changement d'état d'achèvement de l'activité et l'arrêt du monitorage). Chaque effet est décrit sous la forme d'un triplet *(item, propriété, valeur)*, soit la valeur que doit prendre la propriété de l'item. Si cette valeur est d'un type de base (chaîne de caractères, valeur numérique ou booléenne, *etc.*), alors un simple message d'affectation sera adressé à la représentation objet (i.e. l'ItemObject) concerné. Autrement dit, aucune interprétation particulière de l'effet n'est réalisée *a priori*. Si par contre la valeur est elle-même un item, alors la mécanique est légèrement plus conséquente. Le CASO de l'itemObjet représentant la valeur est interrogé pour connaître le sélecteur du message à envoyer à l'itemObjet concerné, pour qu'il y ait application de l'effet. Autrement dit, si le triplet est *(éclairage du salon, état, allumé)*, où *allumé* est un concept manipulable (un item), cela revient à demander à ce dernier s'il sait comment il faut s'y prendre concrètement avec un éclairage pour changer son état dans ce sens. On peut alors supposer que l'itemObjet *éclairage du salon* mettra en oeuvre le patron *proxy*, permettant grâce à ce message l'allumage de l'artefact modélisé. On retrouve la mécanique des requêtes d'IHM, à la différence de forme près que l'objectif présent n'est pas d'interagir avec la personne, mais simplement de lui fournir des services, dans l'esprit des

architectures ambiantes traditionnelles.

6.3.3 Activités domiciliaires complexes

Ce second cas ramène le débat autour de la qualification des erreurs de réalisation. Les activités domiciliaires complexes sont au centre de la discussion depuis le début de ce mémoire, les recettes de cuisine en constituant l'exemple privilégié.

Contrairement aux activités domotiques, une recette de cuisine ne se résume généralement pas en un simple couple tâche/méthode. On conviendra, sans en donner d'exemple, qu'il s'agit d'activités beaucoup plus complexes, tant au niveau structurel que sémantique, supportant de nombreuses contraintes de réalisation (tout particulièrement ordonnancement et gestion du temps), le tout se déroulant dans un espace où les dispositifs potentiellement dangereux sont en nombre. Il faut donc mettre en oeuvre une unité de monitorage qui soit plus complexe.

Le système EpiTalk

L'inspiration a été trouvée du côté des *système conseillers*, avec EpiTalk (notamment [111, 112, 114]). Le principe très général d'EpiTalk est de "brancher" sur une application existante dont on souhaite conseiller l'utilisateur une structure disposant des connaissances nécessaires pour le faire. Pour le système existant, appelé *hôte*, la présence du système conseiller reste transparente, une technique dite d'"espionnage" étant mise en oeuvre pour que le conseiller ait connaissance des actions de l'utilisateur. EpiTalk fonctionne en effet tel un *système épiphyte*[3], dont "l'activité consiste à observer un autre système sans le perturber, et à analyser ces observations pour produire un certain type d'abstractions, que l'on appelle conseils" [110]. La technique d'espionnage repose sur des principes de programmation par objets et tout particulièrement le "détournement" des messages envoyés à un objet du programme hôte.

Pour conseiller l'utilisateur, Epitalk utilise une structure appelée *arbre des tâches*, qui décrit une expertise sur la façon d'utiliser l'application hôte, basée sur les actions de l'utilisateur. Ces actions représentent en effet les noeuds terminaux de la structure arborescente, au dessus desquels différents niveaux d'abstractions sont construits, formalisant ainsi l'expertise et offrant, de bas en haut, une vue locale de plus en plus globale de la

[3]En botanique, nom donné aux plantes qui se fixent sur d'autres sans pour autant se comporter en parasites.

problématique.

Lorsque les noeuds terminaux sont activés, suite à la détection - par espionnage - d'actions, l'information remonte de façon ascendante à travers la structure jusqu'à la racine, à moins qu'un noeud "décide" qu'il ne soit pas nécessaire d'en aviser son parent. Les raisonnements locaux sont assurés par des automates finis, assurant la mise à jour du modèle de la tâche et si nécessaire la production d'un conseil. Au final, l'utilité de la structure est double : (1) analyser le déroulement de la tâche pour connaître les intentions de l'utilisateur, et (2) produire des conseils. D'après les auteurs, si d'autres techniques permettent une analyse plus fine du déroulement, celle-ci oriente l'analyse vers la finalité du système, soit la production de conseil. Le résultat de l'analyse est ainsi directement exploitable [110]. Enfin, on signalera qu'EpiTalk a été pensé comme un système distribué, chaque noeud étant alors vu comme un agent disposant d'une intelligence locale sur la réalisation et capable de prendre les décisions qui le concerne.

EpiTalk et monitorage

L'arbre des tâches EpiTalk offre un support bien adapté au monitorage des activités domiciliaires complexes. Sur le fond, sa structure est très proche du modèle de représentation des activités. Sur la forme, si l'on suppose qu'à un moment donné toute tâche ne peut être réalisée que selon une unique méthode, il faut alors voir un noeud EpiTalk comme un "super" élément, encapsulant la tâche et la méthode sélectionnée, soit le but et le comment de sa réalisation, ce qui est tout à fait cohérent.

Son approche de l'analyse de la réalisation s'adapte parfaitement aux caractéristiques du modèle, avec la possibilité de valider localement les contraintes de réalisation, tout en ayant une vue de plus en plus macro de l'activité. Que l'analyse soit directement orientée vers le conseil est des plus intéressants, même si dans le cas présent c'est la qualification des erreurs et non le conseil qui sera retenue. Quoi qu'il en soit, il est pertinent de pouvoir exploiter directement les erreurs détectées.

Il n'y a par contre aucun besoin particulier d'espionnage dans le cas présent, l'écoute des actions, i.e. des événements contextuels d'interactions, étant parfaitement ouverte. Mais cela ne remet en rien le principe de l'arbre EpiTalk, puisque c'est toujours la survenance de ces actions qui va déclencher le traitement. Néanmoins, on reviendra dans la conclusion de ce chapitre sur une limite du modèle actuel dont la modification pourrait améliorer le côté épiphyte de cette mise en oeuvre s'inspirant d'EpiTalk.

Principe de mise en oeuvre en Java

EpiTalk est, comme son nom l'indique, initialement écrit en SmallTalk. Pour Archipel, une version Java de l'arbre des tâches selon EpiTalk a été écrite, selon le principe suivant : utilisation du patron de conception *composite* [57] pour la structure de l'arbre, utilisation du patron *observateur* pour les outils d'analyse de la réalisation, de qualification des erreurs et autres vues du système.

La patron *composite* permet d'organiser en structure arborescente les objets pour représenter des hiérarchies tout/parties, tout en offrant de l'extérieur la manipulation uniforme des objets composés et individuels (soit les noeuds intermédiaires représentant les tâches abstraites et les noeuds terminaux liés aux actions). On est donc en présence de trois classes : une abstraite représentant le comportement commun de tout élément de la structure, et qui est étendue par deux autres, représentant les éléments terminaux et les éléments composites. Logiquement, c'est la classe commune qui implémentera l'interface MonitoringUnit.

Génération de l'unité de monitorage

Cette structure fait de chacun de ces éléments une unité locale de monitorage de l'activité. Néanmoins, du point de vue du gestionnaire d'unités de monitorage, l'unité référencée se limitera à la racine de l'arbre. Cela implique que la mise en oeuvre assure la transmission dans l'ensemble de la structure des changements réalisés à la racine, notamment lorsque le gestionnaire invite l'unité à changer d'état de monitorage, ce qui se fait parfaitement grâce au patron *composite*.

Du point de vue structurel, l'arbre des tâches n'est pas forcément une copie conforme du modèle de l'activité. Il y a d'une part le concept de méthode qui disparaît, comme cela a déjà été précisé, mais il y a surtout la nécessité pour le monitorage d'une gestion directe des contraintes. L'exemple est le suivant : du côté représentation de l'activité, il est possible pour une méthode donnée de décrire une contrainte partielle de séquence, ne portant donc que sur un sous-ensemble de ses sous-tâches. Côté arbre des tâches, cette contrainte partielle sera isolée, permettant de ne travailler que sur des ensembles homogènes de noeuds (i.e. qui sont soumis aux mêmes contraintes), simplifiant la logique algorithmique sous-jacente. Toutefois, pour conserver la relation ensembliste méthode/sous tâches initiale, un noeud supplémentaire anonyme doit être créé dans la structure, tel qu'illustré à la figure 6.3. C'est au créateur d'unités de monitorage pour les activités domiciliaires

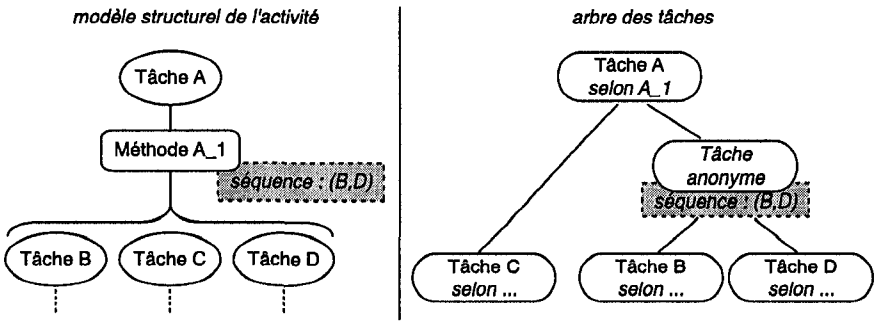

FIGURE 6.3 – Du modèle de l'activité à l'arbre des tâches : prise en compte d'une contrainte de séquence partielle.

complexes qu'incombe cette charge, celui-ci donnant en sortie un arbre des tâches selon EpiTalk.

Traitement des interactions et validation des contraintes

Dans EpiTalk, le traitement des actions "espionnées" se fait selon une approche ascendante, soit des feuilles à la racine. Dans la version mise en oeuvre pour Archipel, ce traitement a été élargi pour permettre une analyse descendante, dont la justification va être donnée dans cette section.

Du point de vue de la mise en oeuvre, chaque noeud de l'arbre est observable par un ensemble d'objets que l'on qualifiera de *contrôleurs*, chargés d'analyser les changements d'état de l'élément pour y détecter les erreurs de réalisation puis les qualifier[4]. Si ce changement concerne l'état de monitorage de l'unité, alors les contrôleurs se limitent à en prendre connaissance, ce qui peut éventuellement entraîner une modification de leur propre statut. Par exemple, un contrôleur de type minuteur, chargé de valider une contrainte temporelle liée à l'initiation, se déclenchera lorsque l'état de monitorage passera à "actif".

Si par contre le changement concerne l'état d'achèvement du noeud, ce qui sous-entend qu'à un bas niveau une action liée à cette réalisation est intervenue, alors le comportement des contrôleurs sera déterminant. En fait, le changement d'état d'achèvement est un processus subjectif, sur lequel les contrôleurs ont droit de regard. Le processus se fait en

[4]D'autres objets, dont la nature n'est pas liée au contrôle, peuvent aussi être observateurs. On pense notamment à la représentation graphique des noeuds de l'arbre dans l'application de suivi de la réalisation.

deux temps :

1. la survenance de l'action est diffusée de niveau en niveau. A chaque niveau les contrôleurs en sont avisés, selon le principe du patron observateur. Ils ont alors la possibilité de mettre leur veto sur le changement d'état, ce qui signifie que de leur point de vue ce changement va à l'encontre d'une bonne réalisation de l'activité, ou plus concrètement viole les contraintes dont ils sont les gardiens. A noter que la présence d'un veto bloque la transmission de l'information au niveau supérieur ;
2. si un veto a été posé, alors les noeuds concernés sont avisés du refus de changement d'état. Dans le cas contraire, c'est la validation du changement d'état qui est diffusée.

On comprend dès lors qu'une action n'entraînera un changement d'état d'achèvement que si celle-ci est valide par rapport aux contraintes de réalisation de l'activité. Dans le cas contraire, le contrôleur va qualifier l'erreur et notifier les observateurs du diagnostic émis. En fait, le diagnostic va remonter dans la structure jusqu'au noeud racine, qui assurera la notification. Cela découle du principe selon lequel, du point de vue du gestionnaire d'unités, seul le noeud racine est connu comme unité de monitorage de l'activité. C'est donc auprès de ce noeud que les objets chargés de récupérer les diagnostics pour les traiter s'enregistreront, lorsque le gestionnaire avisera l'application qu'une nouvelle unité a été créée.

Jusqu'à présent, la logique d'analyse a suivi une approche ascendante, sous-entendant que seule la survenance d'actions terminales, i.e. d'événements d'interactions atomiques, pouvait déclencher le processus d'analyse. Pour Archipel, cette logique doit être poussée plus en avant. En effet, comme cela a déjà été dit, les interactions explicites constituent une source non négligeable d'informations pour le système. Or rien n'impose que l'interaction explicite dénote la réalisation d'une action atomique. Bien au contraire, le système peut être amené à interagir avec le client sur des concepts plus abstraits : c'est la différence entre demander à la personne si elle a sorti les couteaux, puis les fourchettes, puis les cuillères et lui demander plus globalement si elle a sorti les couverts. En fonction des situations et des profils cognitifs, l'une ou l'autre des approches peut s'appliquer. Dès lors, dans le second cas, la question est de savoir comment effectuer l'analyse de cette interaction explicite abstraite.

Le traitement de l'information dans l'arbre des tâches est alors effectuée en deux temps :

1. la validation explicite est descendue du noeud abstrait vers les racines, selon un parcours d'arbre en profondeur suivant la logique des contraintes d'ordonnancement de l'activité[5]. Le principe est d'aller vérifier auprès des noeuds terminaux si cette validation explicite n'est pas en contradiction avec leur état. Il est nécessaire de descendre jusqu'à ce niveau à cause des indices de réalisation qui, s'ils ne témoignent pas de l'achèvement de la tâche, n'en sont pas moins nécessaires. Les indices sont gérés par des contrôleurs au niveau des noeuds terminaux, qui gardent trace de leur survenance sans déclencher le traitement réservé aux événements discriminants[6]. Ces contrôleurs pourront donc s'opposer à la validation explicite si l'indice n'a pas été réalisé ;
2. dans un second temps, si la validation n'a pas été dénoncée par les noeuds concrets, alors l'approche traditionnelle ascendante est mise en oeuvre.

La génération d'un diagnostic peut aussi survenir en dehors de toute interaction. C'est particulièrement le cas avec les contraintes temporelles, où les événements contextuels attendus concernent essentiellement l'écoulement du temps. Ainsi, un contrôleur de ce type peut être amené à générer un diagnostic de façon autonome, sans que la structure ait été soumise au traitement d'une interaction implicite ou explicite. De même, des contrôleurs plus génériques ont été mis en oeuvre, notamment pour monitorer les déplacements d'une personne au cours de la réalisation d'une activité. Ces contrôleurs se basent sur les préconditions spatiales des unités monitorées pour valider les déplacements de la personne, et si nécessaire diagnostiquer un trouble de l'attention.

6.4 Discussion

Avec le monitorage des activités, on aborde une étape déterminante de la sensibilité au contexte d'assistance cognitive, mais aussi des plus délicates. A ce stade du développement, il s'agit clairement de donner au système une capacité de raisonnement, sur une thématique complexe car touchant au fonctionnement humain.

L'option prise pour Archipel est d'offrir au système une connaissance *a priori* de

[5]Un patron de conception *itérateur* est utilisé pour parcourir les sous-tâches d'un noeud. Cet objet itérateur est construit par les contrôleurs concernés, tout particulièrement ceux dénotant une séquence dans la réalisation.

[6]Des traitements peuvent tout à fait avoir lieu lors de la survenance d'un indice. On rentre toutefois dans une logique *a priori* plus complexe : prise en compte du poids de l'indice, indice partagé par d'autres noeuds de l'activité courante, *etc.*

l'activité, décrivant un ensemble de solutions adéquates de réalisation. Ce modèle est ensuite transformé en un arbre des tâches, dans l'esprit d'EpiTalk, pour permettre le suivi de la réalisation à partir des événements contextuels d'interaction perçus. L'objectif est alors de détecter la violation des contraintes de réalisation, que l'on suppose dénoter un trouble cognitif qui sera qualifié par le système, selon une définition très algorithmique de ces troubles. Ce processus vise ainsi à offrir au module chargé de "penser" l'assistance une base concrète de raisonnement, à laquelle on peut facilement associer un profil et des stratégies d'assistance.

La description des activités "à la main" est une tâche délicate et fastidieuse. La mise en place d'une mécanique d'apprentissage vient bien entendu à l'esprit, mais sans rentrer dans une telle discussion, on remarquera que mis à part Opportunity Knocks, l'ensemble des autres orthèses - dont COACH et Autominder - nécessitent cet effort de description. De plus, le cas d'Opportunity Knocks est particulier, puisque les activités qui y sont apprises ne sont "que" des chemins, un modèle offrant différents niveaux d'abstraction sur l'activité semblant nécessaire pour pouvoir (1) décrire avec exactitude les contraintes de réalisation, et (2) offrir à la personne une assistance adaptée à ses besoins, allant du rappel d'un but de haut niveau à la présentation procédurale des tâches concrètes, comme on le verra au prochain chapitre.

Ce modèle, tel que décrit actuellement, témoigne cependant d'une grande limitation : il est dépendant d'un environnement donné. En effet, les informations associées aux méthodes des tâches terminales correspondent aux interactions implicites calculées pour un environnement donné. Cela sous-entend qu'un modèle de recette de cuisine, par exemple, ne pourra être réalisé que pour un environnement donné, ce qui n'a pas de sens. Côté modèle, la prochaine étape est donc d'offrir l'indépendance vis à vis de l'environnement. Pour assurer le pont entre les deux, nécessaire au monitorage, une idée à explorer est de baser la représentation sur les objets de la vie quotidienne et les actions de la personne vis à vis de ces objets. En l'état actuel des choses, si la personne doit prendre un couteau, c'est très certainement l'événement d'ouverture du tiroir contenant le couteau qui sera spécifié, voire une modification de l'ensemble des éléments contenus par ce tiroir si une identification par radio-fréquence est disponible. Idéalement, seule la notion de couteau ou d'artefact abstrait permettant de couper les aliments devrait être nécessaire, couplée à une action telle que "prendre" ou "utiliser", le système devant gérer le contexte pour inférer que telle ou telle interaction correspond à ces caractéristiques. Cela semble tout à fait faisable au sein du *framework* Archipel, dans l'esprit du tra-

vail réalisé pour l'inférence des interactions homme-machine implicites. On disposerait alors de la représentation générale d'une activité, dont un arbre des tâches serait généré spécifiquement pour un environnement donné.

Du côté de la qualification des erreurs, il y aurait un travail conséquent à réaliser pour donner aux solutions algorithmiques proposées une plus grande consistance, voire offrir un modèle d'une tout autre nature pour y parvenir. A noter que des travaux ont été réalisés dans ce sens autour de la modélisation et des architectures cognitives [49].

Pour terminer, l'effort a été mis sur la détection et la qualification des erreurs, prises de façon unitaire, conformément au principe de monitorage de l'activité. Pour aller vers l'assistance, une étape supplémentaire d'intégration est cependant nécessaire, afin d'offrir à l'orthèse une vue plus globale de la situation. En effet, imaginons un scénario très simple où la personne commet plusieurs fois de suite la même erreur. Du point de vue du monitorage, qui tout du moins actuellement met en oeuvre des processus simples automatiques et atomiques, cela va amener le système à effectuer x fois de suite le même diagnostic. Si l'on prend un peu de recul, une solution plus sensée serait d'effectuer une première fois ce diagnostic puis de le conforter, avec pour incidence du côté de l'assistance le renforcement potentiel de l'intervention ou peut être l'essai d'une nouvelle approche. Il s'agit là à nouveau d'un problème des plus subtils qui sera notamment abordé dans le prochain chapitre.

Chapitre 7

archipel.assistance

Après ce long cheminement, qui a conduit le lecteur de la présentation de troubles cognitifs à leur qualification algorithmique, l'ensemble étant vu comme la partie immergée de l'assistance cognitive, il est temps d'entrevoir ce qui va réellement être produit par l'application pour aider la personne à réaliser ses activités domiciliaires.

Dans ce chapitre, c'est l'utilisation du contexte d'assistance cognitive qui est mise en avant, avec la notion d'*acte d'assistance*, que l'on définira dans un premier temps. Cette formalisation permettra de réfléchir à la teneur de la mise en oeuvre de l'assistance dans les orthèses actuelles, et d'en présenter quelques techniques avancées. Puis, c'est le *framework* archipel.assistance qui sera introduit, dédié à la mise en oeuvre des actes d'assistance dans une logique de continuité de la détection et qualification des erreurs. Deux mises en oeuvres de ce *framework*, sous la forme d'actes d'assistance complexes, seront aussi présentées.

7.1 Formalisation de l'assistance : la notion d'acte

Par *acte d'assistance*, on entend toute intervention réalisée par l'orthèse cognitive destinée à répondre à un besoin du client. L'intervention peut être réalisée à la demande explicite de celui-ci, à l'initiative de l'orthèse qui agira alors conformément au profil d'assistance du client, ou selon un mode partagé, l'orthèse répondant à une demande explicite antérieure du client. L'intervention peut concerner le client de façon directe ou indirecte. Dans le second cas, la réponse au besoin se fera par l'intermédiaire d'une intervention auprès d'un tiers (contexte humain) ou d'un élément du contexte matériel. Enfin, l'intervention peut être réalisée selon une modalité synchrone ou asynchrone.

7.1.1 Initiative de l'acte d'assistance

Le client est à l'initiative de l'acte d'assistance lorsqu'il demande explicitement à l'orthèse de lui fournir de l'assistance. Bien entendu, en l'état actuel des développements, la possibilité d'interaction explicite dépend des fonctionnalités d'assistance offertes par l'orthèse, qui elles-mêmes découlent des besoins du client. Dans Archipel, un outil rentrant dans cette catégorie et spécifiquement développé pour répondre à un besoin mnésique est le *localisateur d'objets*, qui permet, à partir d'une sélection adéquate d'objets présentés sur un écran, d'obtenir de l'assistance pour les localiser dans l'environnement, grâce aux effecteurs qui y sont distribués. Une illustration de cet outil sera donnée ultérieurement.

Lorsque le processus de monitorage d'une activité en cours de réalisation est lancé, parce que tel est le profil d'assistance du client ou telle a été sa demande, la production d'un diagnostic ouvrira la porte à un acte d'assistance dont l'initiative est laissée à l'orthèse, dans les limites du profil de la personne. Il s'agit bien entendu du cas le plus emblématique, car symbolisant l'idée d'un dispositif technologique intelligent, tout du moins en apparence. L'aspect autonome de cette initiative provient du fait que c'est l'orthèse qui décide du "quand" et dans une moindre mesure du "comment" de l'acte.

Enfin, si la personne demande à l'orthèse de lui rappeler une activité, l'initiative de l'acte sera partagée : il s'agit bien d'une demande explicite de la personne, mais dont la production sera fonction du CAC, plus précisément du contexte temporel. L'orthèse a alors pour rôle de suivre l'écoulement du temps, et de lancer au moment opportun l'acte.

7.1.2 Intervention directe ou indirecte

Le client reste l'élément central de tout acte d'assistance : c'est pour répondre à ses besoins que l'orthèse agit. Néanmoins, cette réponse peut ne pas concerner directement le client. L'exemple le plus significatif est certainement celui de la cuisinière laissée allumée, et dont le client ne semble pas se soucier. Si l'orthèse décide d'interagir avec le "contrôleur" de la cuisinière pour gérer son extinction, il s'agit clairement d'un acte d'assistance, mais indirect.

Un second cas d'intervention indirecte est l'appel à un tiers, sur lequel il est nécessaire de s'arrêter quelques instants. Si l'envoi d'information à un aidant semble naturel dans un contexte de télé-médecine, le fait pour une orthèse cognitive de devoir faire appel à un moment donné à un autre humain pour aider son client revient à relativiser la capacité d'assistance de ces dispositifs. Or cela est non seulement acceptable mais certainement

indispensable, témoignant finalement de l'honnêteté du développeur vis à vis de la personne assistée. Comme le disait V. Rialle et E. Stip (voir chapitre 2), la technologie va devoir faire face à de nouvelles responsabilités. Une de ces responsabilités est très certainement d'être capable, pour une orthèse, d'arriver à prendre consciense que malgré tous ses efforts, elle n'est plus à même d'aider son client, et que la solution pour assurer le bien-être et la sécurité de ce dernier réside dorénavant dans l'appel à une ressource externe[1]. Faire rentrer l'aidant dans la boucle d'assistance a fait l'objet de travaux du côté des orthèses mobiles, notamment pour MAPS (voir chapitre 2), mais dans un contexte des plus simples puisque l'appel à l'aidant était vu comme la solution par défaut en cas d'erreur (en l'occurrence, lors d'une activité de déplacement). Offrir à l'orthèse le niveau de raisonnement permettant d'aller jusqu'à la solution externe, ce qui suppose qu'en amont elle ait su envisager les différentes pistes possibles sans pour autant arriver au résultat escompté, reste certainement parmi les plus beaux défis de ce type de recherche.

Bien entendu, le cas le plus évident d'intervention est l'interaction directe entre l'orthèse et le client. La transmission de l'information de la première au second peut alors se faire selon différentes modalités et niveaux d'abstraction. Il s'agit clairement de l'approche à privilégier, dans l'esprit des échanges pouvant intervenir entre la personne aidée et un aidant. L'intervention auprès des dispositifs matériels se justifie en dernier recours lorsque la sécurité est mise en jeu (cuisinière allumée, débordement d'eau, *etc.*), si le contexte technologique le permet. Dans les autres cas reste la solution externe.

7.1.3 Intervention synchrone ou asynchrone

Traditionnellement, l'intervention sera synchrone. Dans un contexte de type télémédecine, ou pour signaler certains problèmes non urgents à un tiers, l'orthèse peut être amenée à utiliser un média de communication asynchrone, comme le courrier électronique. Il s'agit toujours d'un acte d'assistance, car l'envoi des informations répond forcément à un besoin exprimé. Cela sous-entend que le comportement de l'orthèse suive une certaine logique éthique.

[1]De tels propos doivent toutefois être relativisés par le niveau d'importance de l'activité pour la personne assistée.

7.2 Actes usuels et techniques de génération d'actes

Cette définition posée, il est intéressant de regarder d'une part quels sont les actes que l'on rencontre de façon usuelle dans la littérature et d'autre part quelles sont les techniques permettant aux orthèses concernées d'aboutir à leur mise en oeuvre.

7.2.1 Orthèses mobiles et actes d'assistance

Les orthèses mobiles, dont les plus significatives ont fait l'objet d'une présentation au chapitre 2, abordent un cercle restreint de thématiques d'assistance : rappel d'activités, guide de réalisation (aide procédurale), aide aux déplacements. Pour nombre d'entre elles, la technique de mise en oeuvre est simple, l'objectif étant d'offrir une réponse directe et synchrone à un besoin explicitement exprimé, sans traitement supplémentaire de la part du système.

Avec Opportunity Knocks et dans une moindre mesure MAPS, l'orthèse bénéficie d'une entrée implicite de données concernant la localisation de la personne dont les déplacements sont monitorés, permettant à l'orthèse de prendre l'initiative de l'acte d'assistance. En effet, grâce à l'utilisation pro-active de ces interactions implicites, l'orthèse est à même d'inférer les erreurs de parcours et de proposer une solution de rechange. La mise en oeuvre de cet acte repose sur une connaissance des ressources locales en matière de transport. Celles-ci sont, pour Opportunity Knocks, du ressort d'un serveur informatique qui effectue les calculs et transmet l'information à l'application installée sur le téléphone cellulaire, qui communique alors avec la personne [151].

7.2.2 Introduction de techniques de planification

Un pas est franchi avec l'orthèse mobile PEAT [90]. Si elle n'offre sur la forme que le rappel d'activités, elle propose sur le fond des outils avancés pour la gestion du quotidien. Ainsi, si l'initiative de l'acte est partagée (le client informe du besoin de rappel, l'orthèse effectue le rappel au moment opportun), elle ne l'est plus de façon équitable. En effet, l'orthèse est dorénavant à même de raisonner sur l'opportunité du rappel, compte-tenu du contexte global de réalisation. Ainsi, si la personne prend du retard dans la réalisation d'une activité[2], le système évalue l'impact de ce retard sur les activités restantes, ce qui revient à résoudre un ensemble de contraintes temporelles. Si un ajustement devient

[2]Cela implique que l'orthèse s'enquiert auprès de la personne de la fin de la réalisation des activités.

nécessaire, l'orthèse base la replanification sur les critères suivants : (1) le caractère "flottant" de certaines activités, pour lesquelles l'utilisateur a explicitement signalé qu'elles pouvaient être temporellement relocalisées sans incidence notoire, et (2) le degré de priorité des activités, critère purement subjectif et ô combien pertinent pour l'organisation du quotidien. Ainsi présentées, les idées mises en avant par cette orthèse[3] semblent des plus pertinentes. Sur la forme, le site internet de la société qui commercialise PEAT[4] fait état d'évaluations cliniques initiées en 2007. Il faudra attendre la publication de ces résultats pour connaître l'utilisabilité réelle de cette orthèse.

Du côté d'Autominder, la planification est utilisée comme un outil d'évaluation *a priori* de la pertinence des rappels [101]. Au cours de la phase d'initialisation de l'orthèse, l'ensemble des activités à réaliser est encodée par un intervenant. Ce plan initial est alors traité pour déterminer quels sont les éléments de ce plan que l'orthèse doit effectivement rappeler. L'idée sous-jacente est la suivante : l'orthèse ne doit donner que le coup de pouce nécessaire pour aider la personne. Autrement dit, rappeler une activité qui n'aurait pas besoin de l'être est une erreur car cela peut entraîner une dépendance du client vis-à-vis du système au lieu de favoriser son autonomie. Pour ce faire, Autominder utilise une technique de planification par réécriture [4], qui amène l'orthèse à repenser son plan de rappel tant que la qualité de celui-ci n'est pas optimale. Le plan initial correspond au pire cas, avec le rappel de toutes les activités inscrites. Un plan voisin est généré de par l'application de règles de réécritures décrivant les besoins de la personne (par exemple, les activités rarement oubliées n'ont pas à être rappelées). Ce nouveau plan est évalué en fonction de critères spécifiques et ainsi de suite jusqu'à l'atteinte d'un seuil de qualité jugé suffisant. On est donc à nouveau en présence d'idées fort intéressantes mais dont l'évaluation auprès de la clientèle manque. On peut d'ailleurs s'interroger sur la complexité de l'écriture des règles de réécritures et des critères d'évaluations du plan de rappel qui soient de qualité, les auteurs ne faisant pas écho de travaux à ce sujet mais se limitant à la présentation de la technique de planification.

[3]L'orthèse MEMOS propose des services proches, mais historiquement l'utilisation de techniques de planification a été introduite par PEAT.

[4]http ://www.brainaid.com/, consulté le 10 juin 2008.

7.2.3 Introduction d'une mécanique de raisonnement probabiliste

Une dernière approche intéressante est celle mise en oeuvre par l'orthèse COACH, dont on se rappellera que le but est d'assister les personnes dans la réalisation de l'activité de lavage des mains (voir chapitre 2 et 6).

COACH base son raisonnement en matière d'assistance sur un processus de décision markovien [22]. Un plan optimal de réalisation de l'activité est initialement généré pour servir de référence au système de guidage. Ce plan précise pour chaque état, i.e. pour chaque situation contextuelle possible, le meilleur message d'assistance à délivrer. Une situation correspond à un positionnement des mains, un état d'écoulement de l'eau, ainsi que la valeur prise par diverses variables reliées au statut de l'activité (dernière étape réalisée, meilleure étape réalisée jusqu'à présent dans le plan, *etc.*) et à l'historique de l'assistance (nombre de messages donnés jusqu'à présent, nombre d'étapes réalisées sans assistance, *etc.*). Pour générer ce plan, le processus de décision markovien est utilisé pour modéliser l'impact d'un message d'assistance donné (leur nombre est égal au nombre d'étapes nécessaires à la réalisation de l'activité multiplié par trois niveaux d'assistance) sur le comportement de la personne, le tout permettant d'évaluer le plus approprié à la situation.

Comme cela a été précisé au chapitre 2, COACH donne des résultats expérimentaux intéressants. Néanmoins, on doit s'interroger sur l'application à grande échelle de cette technique. En effet, le plan initialement calculé pour cette activité relativement simple nécessite, comme le précisent les auteurs, un temps de traitement considérable. Si dans un environnement aussi incertain que celui du comportement humain l'usage de techniques probabilistes ne peut être écarté de prime abord, l'opportunité de faire de ces outils la base du raisonnement en matière de suivi de la réalisation et de génération des actes d'assistance reste à prouver. La solution réside peut être dans l'utilisation au premier plan de techniques moins fines que l'on viendra renforcer lorsque nécessaire pour un traitement de cette nature. Mais il s'agit là d'une question pleinement ouverte.

7.3 *Framework* archipel.assistance

Le *framework* archipel.assistance s'intéresse à la mise en oeuvre des actes d'assistance en général et dans la continuité du processus de monitorage et de qualification sous forme

de diagnostic des erreurs en particulier. Les points abordés concernent la gestion et le suivi des diagnostics, la représentation des actes d'assistance, leur intégration sous forme de stratégie d'assistance et de profil, ainsi que leur gestion.

Comme on le verra, archipel.assistance ne s'attaque à la mise en oeuvre de techniques avancées de gestion de l'assistance (planification, probabilités), mais propose une vue beaucoup plus globale de la thématique. Une fois ce cadre posé, il sera plus aisé de juger de l'opportunité de l'utilisation de ces techniques pour optimiser l'assistance, ces réflexions dépassant le cadre de cette thèse.

7.3.1 Moteur d'assistance

Le moteur d'assistance est le composant[5] chargé de la gestion des actes d'assistance et de leur mise en oeuvre. Le moteur d'assistance s'inscrit comme observateur du gestionnaire d'unités de monitorage. Lorsqu'il a connaissance de la création d'une nouvelle unité, il lui associe une *unité d'assistance*[6]. Parmi les outils à sa disposition, le moteur dispose d'un *régulateur d'actes d'assistance*, et récupère lors de son chargement le *profil d'assistance* du client "principal" (une seule personne étant à l'heure actuelle prise en charge). Les personnes composant le CAC humain sont décrites sous la forme d'ItemObject, le nom d'une classe implémentant l'interface Java AssistanceProfile du *framework* archipel.assistance étant associée à ceux représentant les clients.

7.3.2 Gestion des diagnostics

La gestion des diagnostics est la première étape de l'assistance. Au delà des démarches algorithmiques présentées au chapitre 6, et comme cela a été dit en conclusion de ce même chapitre, il est nécessaire d'offrir à l'application le moyen de prendre un peu de recul sur les diagnostics formulés, qui le sont dans bien des cas de façon très automatique. Il s'agit d'une problématique des plus délicates, dont la mise en oeuvre fera clairement la force d'une telle application.

Dans ce travail, deux aspects ont été abordés : celui de la répétition et de la priorité des diagnostics. Si une personne commet un type d'erreur, il est très vraisemblable qu'elle la commette à nouveau avant de la comprendre et d'y remédier. Il va donc y avoir une répétition de l'erreur pour un même et unique diagnostic, qui n'a pas à être répété mais

[5] Au sens du *framework* archipel.fwk.appmanager, voir annexe E.

[6] On notera que tous ces éléments sont mis en oeuvre sous la forme de processus légers (threads) Java.

plutôt conforté. C'est le problème de répétition. Le problème de priorité se pose pour sa part lorsqu'un diagnostic est formulé alors qu'un premier trouble est en cours d'assistance. La question tourne alors autour de l'impact du diagnostic sur le bien-être et la sécurité de la personne, ce qui revient à déterminer une priorité entre les diagnostics du point de vue de l'assistance pour un profil cognitif donné.

Le *framework* archipel.assistance utilise un *régulateur de diagnostics* pour répondre à ces problèmes. Lorsqu'un nouveau diagnostic est établi par une unité de monitorage, l'unité d'assistance qui lui a été attribuée en est avisée (patron observateur) et requiert alors l'expertise du régulateur, qui seul jugera de l'opportunité de travailler sur ce nouveau diagnostic.

Le régulateur gère les diagnostics grâce à une pile. Il opère selon la séquence algorithmique suivante :

- si la pile est vide, tout nouveau diagnostic est pris en charge ;
- si le diagnostic est identique au diagnostic courant (même type de diagnostic pour la même unité de monitorage), soit celui situé sur le haut de la pile, alors celui-ci est ignoré et l'objet chargé de "manipuler" le diagnostic est avisé que l'approche choisie pour l'assistance ne semble pas avoir fonctionnée ;
- si le diagnostic est de même type que le courant, mais ne concerne pas la même unité de monitorage, alors l'unité courante doit préciser si ce diagnostic rentre sous sa juridiction (problème plus concret pour la même activité). Si c'est le cas, le nouveau diagnostic est ignoré et l'approche choisie signalée comme semblant non fonctionnelle ;
- si le diagnostic est identique à un autre diagnostique de la pile (type et unité), alors il est ignoré et l'assistance pour le diagnostic courant signalée comme semblant non fonctionnelle ;
- ces tests passés, le diagnostic est considéré comme nouveau. Si sa priorité dépasse celle du diagnostic courant, il est mis en haut de pile et son assistance lancée. Sinon, il est placé sur le fond de la pile pour traitement ultérieur.

Un diagnostic est dépilé lorsqu'il est résolu, ce qui déclenche le traitement du diagnostic suivant. La notion de résolution est liée à la survenance d'un événement dénotant la correction de l'erreur ayant entraînée le diagnostic. A titre d'exemple, si un séquence de réalisation a été violée, c'est l'interaction normalement attendue d'après la séquence qui servira de repère pour le système.

Il se peut que dans certains cas la résolution n'intervienne pas. Dans la version ac-

tuelle, deux situations témoignent de cette non résolution : (1) l'incapacité pour l'orthèse de mettre en oeuvre la stratégie d'assistance, et (2) la mise en oeuvre de cette stratégie n'a pas conduit à la survenance de l'événement dénotant la résolution. L'incapacité témoigne d'une impossibilité technologique, aucun effecteur n'étant à même de fournir le service souhaité[7]. Dans tous les cas, ce sont ces situations qui amènent actuellement l'orthèse à faire appel à une aide extérieure. Dans les faits, cet appel se résume pour l'instant à une notification visuelle via l'interface "développeur" de l'application.

Du point de vue de la mise en oeuvre, on notera que ce *framework* utilise abondamment les patrons de conception comportementaux *état*, *stratégie* et *observateur* [57], déjà définis dans ce mémoire.

7.3.3 Profil d'assistance

Lorsque le traitement d'un diagnostic est entériné par le régulateur, l'unité d'assistance s'enquiert auprès du profil d'assistance de l'existence d'un *gestionnaire* pour ce type de diagnostic. Le profil d'assistance précise aussi le *comparateur de diagnostics*[8] applicable à la personne, ainsi que la *stratégie de changement d'état de monitorage*, qui dénote le principe d'une assistance par défaut, hors réalisation d'erreurs. Enfin le profil d'assistance précise le *niveau d'abstraction* initial applicable en cas d'assistance procédurale.

Gestionnaire de diagnostic

Un gestionnaire de diagnostic fédère l'information d'assistance pour un type de diagnostic. Cette information se présente sous la forme d'un ou plusieurs actes d'assistance, alors organisés selon une *procédure d'assistance*. Par procédure on entend une suite d'actes dont la logique suit *a priori* une progression dans l'assistance : concrétisation des informations fournies, changement de média pour une présentation plus explicite de l'objectif à atteindre (par exemple, une vidéo à la place d'une image), *etc.* La notion de procédure d'assistance s'inscrit clairement dans une démarche de non-dépendance de la personne vis-à-vis de l'orthèse : l'assistance fournie dans un premier temps est minimale, elle ne fait qu'orienter la personne dans la direction adéquate. C'est donc à la personne que revient la décision finale. Toutefois, si la conséquence de l'acte n'est pas celle attendue, l'incitation effectuée par l'orthèse pourra se faire plus concrète. Ce principe d'"augmentation"

[7] Ce qui n'empêche pas la personne d'éventuellement s'auto-corriger ...
[8] Simple outil utilisé pour comparer la priorité des diagnostics.

du niveau d'assistance est actuellement mis en oeuvre lorsque le gestionnaire est avisé du non fonctionnement supposé[9] de l'assistance réalisée. Si nécessaire, une mécanique événementielle permet au gestionnaire d'aviser le régulateur de diagnostic de son "incompétence" vis-à-vis du problème rencontré par la personne.

Régulateur d'actes d'assistance

Le régulateur d'actes d'assistance centralise les actes que les gestionnaires de diagnostics souhaitent mettre en oeuvre. Il agit à la fois comme un filtre et un temporisateur, fonctionnant selon un principe de file d'attente. En tant que filtre, il met de côté les actes dont un équivalent est déjà présent dans la file. En tant que temporisateur, il s'assure qu'un temps respectable intervienne entre deux actes d'assistance.

Lorsqu'il accepte la mise en oeuvre d'un acte, le régulateur demande tout d'abord son initialisation. Par exemple, un acte d'assistance dont l'objectif serait le rappel du but en cours, doit pouvoir trouver auprès de l'activité supportée la description de ce but, dans un format adéquat. Si cette information n'est pas disponible, l'acte manifestera son incapacité à être réalisé et sera mis de côté par le régulateur. Cela peut clairement remettre en cause la capacité de l'orthèse à supporter le client suite à un diagnostic. Concrètement, cela veut dire que le modèle d'activité sous-jacent doit être complet, tant au point de vue du monitorage (expression des contraintes liées à sa réalisation) qu'au point de vue de l'assistance. Dans les faits, les informations d'assistance attendues pour une tâche sont : (1) une présentation textuelle, (2) une illustration visuelle, (3) une illustration audio, et (4) une illustration vidéo, qu'un objet dit *support d'assistance* est chargé de véhiculer. S'agissant des activités domiciliaires complexes, c'est l'arbre des tâches inspiré d'EpiTalk qui joue se rôle, en plus de celui d'unité de monitorage. Sa structure composite permet aisément de jouer sur la possible factorisation des informations au niveau du modèle de l'activité (par exemple, une description audio partagée par tout un sous-arbre des tâches). A noter qu'il est d'ailleurs tout à fait possible d'ajouter dans la représentation d'une activité des noeuds dont le rôle se limite au support d'informations d'assistance. Grâce à la représentation sémantique des tâches, ces noeuds pourront être ignorés du point de vue de la structure de l'activité lors d'une présentation à l'utilisateur (voir présentation de l'assistance procédurale dans cette section).

Lorsque l'initialisation d'un acte est réalisée sans encombre, le régulateur peut enchaîner sur sa mise en oeuvre. Une mécanique d'observation permettra au second d'être

[9]Supposé dans le sens où la personne n'a pas réalisé suite à cet acte l'action initialement attendue.

avisé du bon fonctionnement du premier. Par "bon fonctionnement", on entend l'effectivité de la mise en oeuvre matérielle de l'acte. Si l'acte se manifeste par une interaction machine-homme, c'est la résolution de la requête d'interaction qui témoignera de cette effectivité. Plus précisément, il faut qu'un effecteur puisse fournir, pour cet environnement, le service d'IHM envisagé, tout particulièrement dans les conditions spatiales définies (voir chapitre 5). Néanmoins, si cette interaction suppose une rétroaction de la part de l'utilisateur, alors celle-ci dénotera possiblement à la fois la compréhension de l'acte et la correction de l'erreur qui en est à l'origine. Toutefois, ces deux aspects sont traités séparément par le *framework*, l'un du côté suivi de la réalisation matérielle des actes et l'autre du côté suivi des diagnostics.

Enfin, s'agissant de la temporisation des actes, l'idée est de ne pas submerger la personne d'informations, mais bien au contraire de poser chaque acte l'un à la suite de l'autre, en laissant un temps minimal de repos entre chaque. Si une rétroaction est par exemple attendue, l'acte suivant - si tel est le contenu de la file des actes - ne sera formulé qu'après validation de cette rétroaction à laquelle un temps arbitraire est ajouté. Bien entendu, il y aurait fort à réfléchir autour de stratégies de temporisation et de délivrance des actes, en prenant en compte divers critères comme par exemple le contexte spatial. Le système serait alors amené à organiser les actes en fonction de l'emplacement courant de la personne. Il s'agit à nouveau d'une problématique de recherche à part entière.

Stratégie de changement d'état de monitorage

Pour terminer, le profil d'assistance précise quelle est l'approche à appliquer lorsqu'une unité de monitorage change d'état. Cela revient à exprimer si une assistance par défaut, hors contexte d'erreur, doit être mise en oeuvre lorsque l'unité commence le suivi de la réalisation. Le cas le plus typique est, pour les activités complexes, l'aide procédurale (voir section suivante). La stratégie de changement d'état représente donc un premier niveau d'assistance qui, si nécessaire, sera vraisemblablement complété par des actes ciblés liés aux erreurs de réalisation ponctuelles.

7.3.4 Deux exemples d'actes d'assistance

L'aide procédurale et l'aide mnésique à la localisation des objets constituent les mises en oeuvre du *framework* d'assistance les plus notoires de ce travail. Si la première est bien connue des orthèses cognitives, sa mise en oeuvre dans le cadre d'Archipel est améliorée

de par la richesse du modèle d'activité sous-jacent. De son côté, l'aide à la localisation est une proposition propre à ce travail pour répondre à certains problèmes mnésiques.

Aide procédurale

Par aide procédurale, on entend la présentation étape par étape (procédure) d'un savoir-faire, dans l'esprit du patron "guide" de Maciuszek [98] (voir chapitre 2). Le Visual Assistant d'AbleLink[10] propose une telle fonctionnalité, basée sur l'illustration visuelle des étapes et un simple bouton pour la navigation, pour le passage à l'étape suivante. Mais si la présentation de l'information est une problématique en soit, c'est sur la modélisation de la procédure que l'on discutera ici.

Pour le Visual Assistant, une procédure est une simple séquence d'étapes, ce qui sous-entend une navigation linéaire (précédente, suivante). Dans [113] est présenté un outil spécifiquement dédié à l'aide procédurale. Ce travail de 1997 qui n'a pas connu de suite introduit une notion importante : si une étape est un instantané d'une partie de l'activité, celle-ci peut l'être à différents niveaux d'abstraction. Une étape peut donc cacher un ensemble de sous-étapes plus concrètes, dans l'esprit d'une représentation arborescente de l'activité. Le modèle présenté dans [113] spécifie dès lors pour chaque étape quel est le sous-ensemble de remplacement possible, le temps et l'absence de validation de l'étape courante servant de référence pour modifier le niveau de présentation de l'information à l'utilisateur.

Dans Archipel, c'est l'arbre des tâches généré à partir du modèle de l'activité qui sert de support pour la génération de l'assistance procédurale. Une procédure est belle et bien modélisée comme une séquence d'étapes. Néanmoins, pour une activité donnée, il n'y a pas d'*a priori* sur le contenu de la procédure, et celle-ci peut évoluer au cours du temps. Une procédure est avant-tout une considération sémantique, son contenu évoquant l'activité selon un certain niveau d'abstraction, qui peut être modulé selon les parties de l'activité. Grâce à archipel.lang, une représentation sémantique des tâches est associée au modèle structurel. A chaque tâche est associée une sémantique, elle même décrite sous la forme d'un itemObjet. Des relations d'abstraction sont établies entre ces itemObjets, permettant en combinant structure et sémantique un parcours plus riche de l'activité. Bien entendu, il déjà été dit que la structure arborescente des activités dénotait, en se déplaçant de la racine vers les feuilles, une vue de plus en plus concrète du modèle. Néanmoins, cela n'implique pas que pour une même activité, le même niveau de sémantique soit trouvé au

[10]http ://www.ablelinktech.com/_handhelds/visualassistant.asp

même niveau de profondeur dans toutes les branches de l'arbre, d'où la nécessité d'une représentation sémantique complémentaire. De plus, cela permet d'introduire, dans la structure, des noeuds dont le seul rôle est par exemple de fédérer une information d'assistance commune. C'est alors la sémantique de ces noeuds qui permettra de les manipuler adéquatement, puisqu'ils n'ont notamment pas vocation à être intégrés à une procédure.

Le *framework* archipel.assistance propose les classes nécessaires à la manipulation de l'aide procédurale, toujours de façon abstraite par rapport à un modèle de représentation donné, une mise en oeuvre concrète étant réalisée pour l'arbre des tâches inspiré d'EpiTalk. A noter que chaque étape d'une procédure étant liée à une unité de monitorage, le patron *itérateur* [57] utilisé pour parcourir les étapes tient compte de leur réalisation. Il faut en effet voir que la génération d'une procédure, même si réalisée pour une abstraction donnée (initialement précisée par le profil d'assistance), reste subjective : si la structure de l'activité témoigne d'une séquence de réalisation, alors la procédure la suivra. Dans le cas contraire, la procédure proposera dans la mise en oeuvre actuelle un ordre sans fondement particulier[11]. Cela sous-entend que la personne peut à sa guise, sans commettre d'erreur, anticiper une étape qu'elle sait applicable. L'étape disparaîtra alors de la procédure, tout du moins du point de vue de l'utilisateur. De même, la réalisation d'une étape reliée à un événement d'IHM discriminant entraînera le passage automatique à l'étape suivante. Enfin, on comprendra que lorsqu'une requête d'IHM est formulée pour ce type d'acte d'assistance, le contenu de la requête correspond à l'objet modélisant l'aide procédurale.

Aide mnésique à la localisation à des objets

La recherche d'objets dans l'environnement est une constante chez bien des individus. Clés et lunettes font partie des objets qui bizarrement disparaissent sans arrêt, ce qui a d'ailleurs poussé certains industriels à développer des produits dédiés pour ce marché (porte-clés répondant au sifflement, *etc.*).

Pour réaliser ses activités domiciliaires, tout particulièrement côté cuisine, l'homme utilise un grand nombre d'outils et de composants (ustensiles, ingrédients, *etc.*) qui représentent tout autant de tâches orientées vers la localisation. Du point de vue cognitif, cela nécessite des ressources mnésiques considérables qu'il peut s'avérer nécessaire de supporter. C'est l'objectif de la mécanique d'assistance proposée ici, et qui a été en

[11] On pourrait imaginer la prise en compte de critères divers, tels que la localisation de l'étape, pour construire des procédures qui semblent plus naturelles du point de vue de l'utilisateur.

partie réalisée par deux étudiants de premier cycle universitaire en informatique, dans le cadre d'un cours de type projet.

L'idée est, selon un mode d'interaction explicite, de proposer à l'écran une *page* de vignettes représentant des objets de la vie quotidienne (OVQ), un acte d'assistance de mise en valeur de l'objet étant produit lors de leur sélection. Ces vignettes peuvent en fait représenter deux types d'OVQ : soit il s'agit d'un objet concret et l'aide est immédiate, soit il s'agit d'une catégorie d'objets et la sélection amène à la consultation d'une sous-page d'OVQ. Le travail des étudiants a donc porté sur l'organisation de cette connaissance, sur un outil graphique pour sa constitution ainsi que sur une interface web pour son utilisation côté client. A noter que la connaissance en tant que telle est déjà présente au sein du système, puisqu'il s'agit de la représentation du contexte matériel en archipel.lang. Ce que permet cet outil, c'est une organisation plus simple de celle-ci, adaptée aux besoins de la personne et faisant fi des considérations ontologiques nécessaires à sa manipulation informatique. Le modèle de page se limite donc à faire le lien vers les items, auprès desquels peut être trouvée l'information pour les illustrer (vignette) et pour les localiser. Grâce au "glisser-déposer", il est des plus simples de créer à partir de la représentation du contexte matériel des pages de vignettes, stockées en XML. Elles peuvent aussi être associées à un élément du modèle de l'activité, faisant ainsi le lien entre ces connaissances[12]. Ce type d'aide peut tout aussi bien être intégré dans une procédure d'assistance (et non une assistance procédurale). Le contenu de la requête d'IHM correspondra alors au modèle objet des pages d'OVQ. Une vue de l'outil graphique est présentée à la figure 7.1. L'interface réservée au client est visible dans le prochain chapitre, qui porte sur les mises en oeuvre du prototype.

7.4 Conclusion

Dans ce chapitre, une formalisation de l'assistance cognitive et le *framework* correspondant ont été présentés. Autour du profil d'assistance, les actes d'assistance viennent répondre aux diagnostics formulés côté monitorage, pour aider la personne à assurer la réalisation de son activité et d'une manière plus large son bien-être et sa sécurité. En tant que pointe visible du processus d'assistance cognitive, l'acte repose logiquement sur l'ensemble de la mécanique sous-jacente : représentation des connaissances, interaction

[12]On pourrait automatiquement créer les pages de vignettes si les activités étaient à un bas niveau liées aux objets et non aux actions de la personne. Voir discussion au chapitre 6.

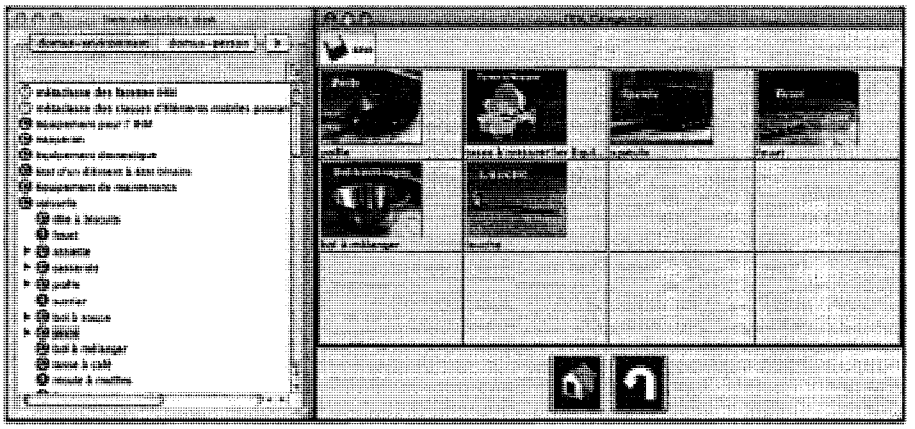

FIGURE 7.1 – Outil graphique pour la création de pages d'objets utilisées comme support pour l'aide à la localisation

homme-machine, monitorage et représentation des activités. Il clôture ainsi la boucle de sensibilité au contexte d'assistance cognitive.

L'outil de localisation des objets illustre les capacités du *framework* pour créer des dispositifs d'assistance. De l'interface graphique de construction des pages d'objets à la production des actes au sein même du cadre de réalisation des AVQ, c'est l'ensemble de la mécanique Archipel qui est utilisée et personnalisée pour répondre à un problème ciblé. Néanmoins, il reste fort à faire autour de la notion d'assistance, la gestion actuelle étant relativement brute. Ainsi, une personne ne respectant pas une séquence de réalisation va être invitée à réaliser la prochaine étape de cette séquence. Si sur le fond le principe semble judicieux, il souffre de profondes limitations dans la forme : qu'en est-il des modifications apportées par erreur à l'environnement ? Si l'erreur s'est matérialisée par la mise en route de la cuisinière, alors que les ingrédients n'étaient pas prêts, ne faudrait-il pas tout d'abord s'assurer de l'extinction de cette cuisinière avant de guider la personne vers la fin de la préparation des ingrédients ? Oui, bien entendu, ce qui pose comme à chaque fois le besoin pour l'orthèse de prendre du recul par rapport aux événements réalisés. Et à nouveau il semble pertinent pour cela de s'intéresser à la nature des objets manipulés, le cas de la cuisinière (et autres appareils classés comme dangereux pour un profil donné) demandant clairement une attention particulière.

Enfin, quelle que soit la force des techniques de raisonnement mises en oeuvre, c'est le facteur humain qui déterminera la justesse de l'assistance fournie. Ainsi, l'expérimentation présentée au prochain chapitre, dont les résultats sont pertinents, illustre tout de même

la réelle difficulté de ce travail : trouver la bonne interaction au bon moment étant donné le profil particulier de la clientèle ciblée.

Chapitre 8

Déploiements et expérimentation d'Archipel

La présentation des déploiements et expérimentation réalisés d'Archipel vient clore la description de ce travail de thèse. Dans ce chapitre sont détaillés les résultats techniques et d'utilisabilité de l'orthèse, ouvrant la porte à une discussion générale sur la recherche accomplie.

Dans un premier temps, une illustration scénarisée du fonctionnement d'Archipel est présentée. Quelques planches mettent en images, à travers différentes vues de l'application *côté développeur* et *côté client*, la mécanique globale d'Archipel pour l'aide à la réalisation d'activités complexes.

Puis il est question des déploiements réalisés hors de la plate-forme DOMUS. Le premier a pris lieu chez le partenaire industriel du laboratoire pour ce projet, et met en avant les capacités d'adaptation du *framework* pour l'utilisation de services de gestion de contexte externes. Le second déploiement est moins conventionnel, puisqu'il s'agit d'un environnement de démonstration mobile construit autour d'une cuisinette jouet, baptisé DOM*ini*US. Par cet outil portable et intégré, on illustre l'essentiel d'une orthèse élaborée autour du paradigme d'informatique ubiquitaire.

Enfin, c'est l'expérimentation du prototype déployé à l'appartement-laboratoire DOMUS et réalisée fin 2007 par douze adultes présentant une déficience intellectuelle qui est décrite. Sont aussi présentés les résultats liés à la phase préparatoire de cette expérimentation, issus d'échanges réalisés avec les intervenants du Centre Notre-Dame de l'Enfant de Sherbrooke, dédié à la clientèle ayant une déficience intellectuelle.

8.1 Archipel, une illustration

Pour terminer cette présentation d'Archipel, il semblait opportun de présenter en quelques planches une illustration du fonctionnement de l'orthèse, incluant l'ensemble de ses fonctionnalités. A partir de quelques images tirées d'une vidéo présentée à la conférence ICTA en juin 2007 [122], des copies d'écran de l'application agrémentées de quelques commentaires, dans un scénario proche de celui décrit au chapitre 3, on illustre Archipel *côté développeur* et *côté client*. Le côté développeur propose des vues de la plupart des outils cités dans ces derniers chapitres. Côté client, on parle d'effecteurs de nature diverses répartis dans l'environnement. L'environnement en question est l'appartement du laboratoire DOMUS, parfaitement fonctionnel et entièrement dédié à la recherche portant globalement sur les habitats intelligents. Les planches sont présentées aux figures 8.1 à 8.3. Des remarques seront formulées ultérieurement concernant la mise en oeuvre des interfaces web.

FIGURE 8.1 – Archipel, une illustration - planche 1

FIGURE 8.2 – Archipel, une illustration - planche 2

FIGURE 8.3 – Archipel, une illustration - planche 3

8.2 Déploiements réalisés hors plate-forme DOMUS

8.2.1 Déploiement chez le partenaire industriel

Dans le cadre du partenariat établi entre le laboratoire DOMUS et France Telecom R&D, qui a apporté son soutien financier à ce projet, Archipel a été déployé au centre de recherche de France Telecom situé dans la région de Grenoble, en France.

L'intérêt de ce déploiement est clairement de confronter Archipel à de nouvelles sources d'informations en ce qui concerne les interactions homme-machine implicites. France Telecom dispose en effet d'outils maison pour l'intelligence ambiante, s'attaquant notamment dans son projet AmiLoc à la gestion des informations de localisation et d'identification des éléments environnementaux (objets et personnes) [55]. Plus précisément, France Telecom a équipé un buffet d'un système de localisation par radio-fréquence pour les éléments de vaisselle, et utilise une mécanique de vision par caméra couplé à une identification par radio-fréquence des individus pour suivre les déplacements de ces derniers dans l'environnement. L'ensemble de ces informations est disponible auprès d'AmiLoc, par une mécanique de requête sur HTTP formalisée en XML.

Dans la version actuelle d'AmiLoc, l'obtention des informations de localisation se fait par sondage (*polling*) du serveur, autrement-dit en adressant de façon régulière une requête de localisation à AmiLoc. Pour ce faire, un processus *ad hoc* a été mis en oeuvre, facilement intégré à Archipel grâce au *framework* archipel.fwk.appmanager (voir annexe E). Du côté représentation des connaissances, il a fallu associer aux classes d'items localisables *via* AmiLoc leur identifiant pour ce serveur, indispensable à l'écriture des requêtes. Un concept représentant les "éléments AmiLoc" a simplement été ajouté et les relations d'implémentation adéquates établies. S'agissant de la localisation par caméra, une représentation virtuelle de ce type de capteur a été introduite. En effet, comme précisé au chapitre 5, Archipel associe un capteur à un item source, les données provenant du capteur étant sémantiquement associées à cet item. Concrètement, si l'on parle localisation, un capteur fournissant un service de ce type sera associé à un item représentant une zone de localisation dans l'environnement (une pièce par exemple). Ainsi, trois capteurs virtuels ont été décrits pour permettre la localisation du client dans les trois pièces de l'environnement, bien que réellement ce service soit fourni par deux caméras.

Il s'agit là des ajouts nécessaires et suffisants pour utiliser le service AmiLoc dans le contexte d'Archipel. On souhaite ainsi illustrer la capacité d'ouverture du *framework*, tout du moins en ce qui concerne la gestion des événements d'interaction, l'ensemble restant

parfaitement transparent pour les services de plus haut niveau (monitorage, assistance).

France Telecom a de plus fourni un prototype d'effecteur sans-fils baptisé "chenillard"[1], utilisant le protocole de communication zigbee[2]. Disposant d'une librairie Java adéquate pour la communication, l'intégration de ce type d'effecteurs a pu être réalisée sans problème, selon la technique du patron *proxy* présentée au chapitre 5. Il s'agit d'une preuve de concept très intéressante, dans la mesure où la mise en oeuvre de solutions sans fil est largement discutée lorsqu'il s'agit de penser au déploiement d'orthèses dans des environnements domestiques existants. De même que précédemment, tout cela a pu être réalisé de façon transparente pour le reste de l'application, et repose essentiellement sur l'ajout de connaissances au niveau de la représentation de l'environnement.

8.2.2 L'environnement mobile de démonstration DOM*ini*US

Un second déploiement a été réalisé hors plate-forme DOMUS, dans le but de pouvoir démontrer en tout lieu les principes d'Archipel et tout particulièrement sa philosophie de l'interaction homme-machine.

Pour ce faire, un environnement mobile de démonstration a été réalisé et exposé pour la première fois à la conférence MobiHoc, organisée par l'ACM en septembre 2007 à Montréal [14]. Cet environnement, dont une illustration est proposée à la figure 8.4, se présente sous la forme d'une cuisinette jouet équipée de divers capteurs, effecteurs, et des contrôleurs nécessaires à leur fonctionnement. D'une hauteur de 120 cm sur 80 cm de large, cette cuisinette est intégrée dans deux malles qui s'adaptent l'une sur l'autre pour la démonstration, ne nécessitant que le branchement de trois connecteurs entre ces deux éléments. Une simple alimentation électrique et le lancement de l'application permettent de faire fonctionner le tout, offrant une solution intégrée pour l'illustration des principes d'Archipel autour d'activités de recette de cuisine[3].

Dans la mesure où cet environnement s'inspire du déploiement réalisé dans l'appartement-laboratoire DOMUS, il a été baptisé DOM*ini*US, pour "mini" DOMUS. Et bien que la simulation d'une recette complexe de cuisine y soit difficile, il reste parfaitement possible, grâce à quelques exemples simples, d'illustrer les principaux concepts de l'assistance ubiquitaire :

[1] Il s'agit d'une rangée de diodes électroluminescentes s'allumant tour à tour pour former un signal lumineux "ondulatoire".

[2] http ://www.zigbee.org/

[3] Mise en oeuvre réalisée avec l'aide de Francis Bouchard côté menuiserie et de Nicolas Marcotte côté électronique. Le tout sur une idée de Sylvain Giroux.

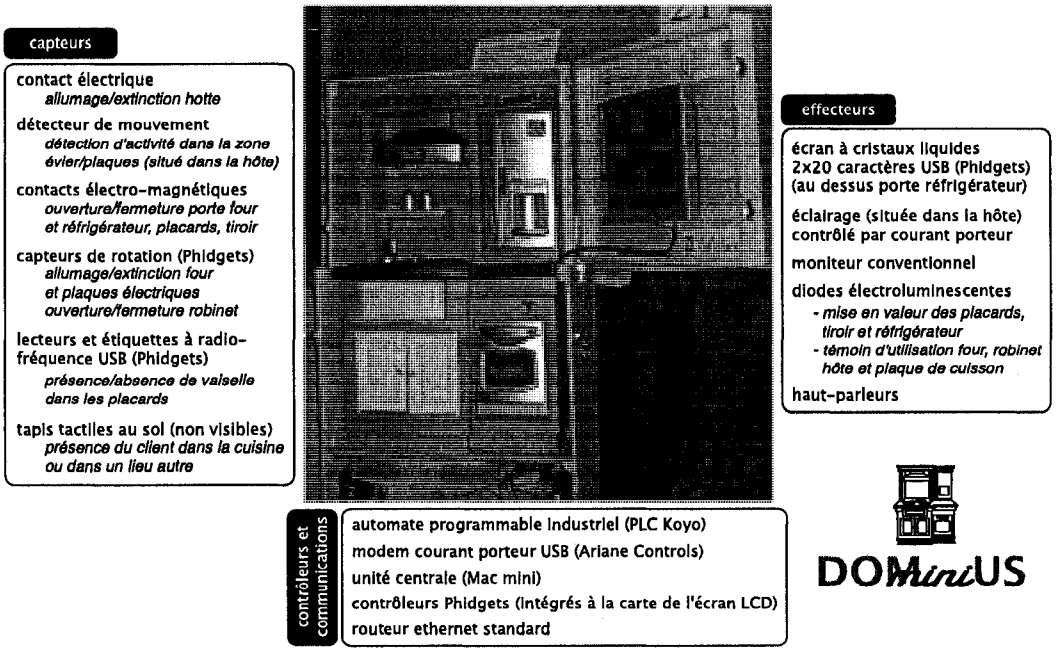

FIGURE 8.4 – L'environnement mobile de démonstration DOM*ini*US - photographie prise à MobiHoc'07

Utiliser l'information en provenance de l'environnement La figure 8.5 présente deux vues SVG[4] commentées de DOM*ini*US. Plus précisément, les vues concernent l'état des itemObjets représentant les capteurs (partie gauche) et les éléments mobiliers (partie droite) de cet environnement. L'API Batik d'Apache[5] permet de manipuler les sources SVG dans un contexte Java. Il est donc aisé de faire de ces éléments graphiques une vue des itemObjets décrivant DOM*ini*US, plus précisément une vue de la propriété représentant leur état. Un changement de couleur de l'élément graphique viendra alors souligner le changement d'état sous-jacent. Ainsi, il est possible de manipuler physiquement l'environnement, en montrant comment l'information issue de capteurs peut être interprétée au niveau logiciel pour inférer les actions réalisées par la personne. C'est toute la mécanique d'interprétation des événements implicites d'IHM présentée au chapitre 5 qui prend ainsi vie. Comme toujours, cette application SVG est intégrée à Archipel grâce au *framework* archipel.fwk.appmanager (voir annexe E).

[4] *Scalable Vector Graphics*, standard XML de représentation vectorielle.
[5] http ://xmlgraphics.apache.org/batik/

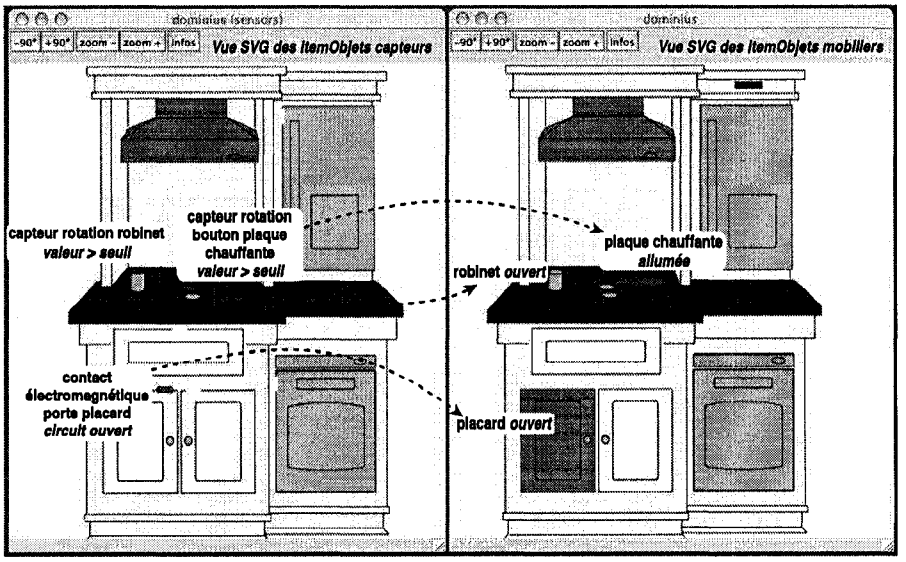

FIGURE 8.5 – Inférence des IHM dans DOM*ini*US : des capteurs aux éléments mobiliers (vues SVG de l'état des itemObjets)

Utiliser l'environnement comme support d'interaction machine-homme et d'assistance

Autour de quelques mini-scénarios, on illustre la relation d'assistance : aide à la localisation des objets avec mise en valeur des emplacements (grâce aux diodes disposées sur les éléments mobiliers), détection d'erreurs lors d'une séquence illustrant un rappel audio ou l'utilisation de vidéos explicatives, principe d'assistance procédurale. Un scénario dont la mise en scène illustre avec efficacité IHM et assistance est le robinet que l'on laisse couler. Au bout d'un certain temps l'orthèse (dont le comportement peut être suivi *via* les différentes vues proposées : arbre des tâches, régulateurs, *etc.*) se manifeste auprès de la personne en faisant clignoter l'éclairage situé dans la hotte (contrôlé par courant porteur). Cela permet d'illustrer (1) le raisonnement du système en terme de validation de contraintes et (2) la mise en oeuvre d'une assistance basée sur les différentes caractéristiques contextuelles. En l'occurrence, l'orthèse a dû faire le lien entre un problème situé au niveau du robinet et un effecteur capable d'apporter l'information d'assistance adéquate pour cette zone d'activité (éclairage hotte). Enfin, il a fallu que le système infère que la détection d'activité dans cette même zone (grâce au détecteur de mouvement lui aussi situé dans la hotte) dénotait la réception du message d'assistance, ce qui devait déclencher l'extinction de l'éclairage.

En conclusion, DOM*ini*US offre une illustration plutôt convaincante de la démarche technique sous-jacente, lorsque présentée à des personnes du milieu technologique. Mais un objectif tout aussi louable serait l'utilisation de DOM*ini*US comme "outil pédagogique", à destination non plus des techniciens mais des utilisateurs potentiels (en incluant tout particulièrement les intervenants en réadaptation, que l'on supposera non-techniciens). L'idée serait alors d'utiliser l'outil pour faire comprendre en quoi ce type de technologie peut apporter une plus-value dans l'accompagnement des personnes souffrant de troubles cognitifs, améliorant leur autonomie tout en soulageant les tiers. Si un grand nombre de personnes du milieu ouvre doucement mais sûrement la porte à la technologie - c'est le cas des intervenants ayant permis l'expérimentation d'Archipel - il s'agit d'une modification conséquente des approches en réadaptation. Les interrogations sont donc forcément immenses. Et il est clair que cette recherche appliquée n'a aucun intérêt si elle est refusée par les principaux intéressés. Dès lors, avoir offert avec DOM*ini*US et Archipel des outils pour une meilleure compréhension de l'utilisation de la technologie à des fins d'assistance cognitive constitue un résultat pertinent, qui, on l'espère, se confirmera dans l'avenir.

8.3 Expérimentation du prototype déployé au Laboratoire DOMUS

Sous la houlette de Dany Lussier-Desrochers, chercheur en psychologie[6], et avec la participation des intervenants et surtout des membres du Centre Notre Dame de l'Enfant de Sherbrooke, a pu être menée à l'appartement-laboratoire DOMUS une première expérimentation d'Archipel.

Dans une recherche interdisciplinaire comme celle-ci les compétences se trouvent tant au niveau technique qu'au niveau de la connaissance de la clientèle. Cette expertise, que détiennent notamment les intervenants en réadaptation, a pu être appréciée lors de la première phase de l'expérimentation. Celle-ci a permis d'une part de concevoir un protocole expérimental adapté et d'autre part d'améliorer les outils d'assistance. Le tout a ouvert la porte à une phase d'expérimentation réelle du prototype par douze adultes présentant une déficience intellectuelle légère, réalisée d'octobre à décembre 2007. Le projet expérimental a été présenté dans [122], [123] (version étendue du précédent),

[6]Dany Lussier-Desrochers est aujourd'hui Professeur au département de psychoéducation de l'Université de Trois-Rivières, (QC), Canada. Il était auparavant post-doctorant au Laboratoire DOMUS. Il est titulaire d'un doctorat en psychologie.

et [95].

8.3.1 Phase pré-expérimentale

La préparation de l'expérimentation nécessitait (1) de présenter aux intervenants la recherche pour pouvoir identifier les personnes susceptibles d'y participer, (2) de définir le type d'activité que les personnes participantes pourraient réaliser avec l'aide de la technologie, et (3) de valider auprès des intervenants les outils d'assistance envisagés.

Il a été facile pour les intervenants du groupe de travail d'identifier parmi leurs bénéficiaires celles et ceux dont l'autonomie au domicile pourrait être améliorée par la technologie. Il s'agit pour la plupart de jeunes adultes vivant en famille ou en foyer, pour qui l'accès à une vie indépendante ne nécessiterait qu'une faible amélioration de leur autonomie. De par leurs activités au centre, tous disposent d'un minimum de connaissance en cuisine, mais peu d'entre eux la pratique au quotidien. Côté technologie, la plupart ont pu expérimenter l'utilisation d'un ordinateur par le biais de recherches sur internet. Seuls quelques-uns disposent d'une machine à leur domicile.

Le choix de recettes de cuisine comme type d'activité pour l'expérimentation s'est naturellement imposé. La préparation des repas est à la fois un incontournable de la vie autonome et une activité potentiellement complexe et dangereuse. Le centre propose d'ailleurs à ses bénéficiaires des cours de cuisine, dont l'objectif est d'une part de leur faire acquérir des réflexes pour la structuration de ce type d'activité et d'autre part d'apprendre un ensemble de recettes. Les intervenants ont développé une méthodologie d'apprentissage spécifique que les chercheurs ont pu découvrir au cours de la phase pré-expérimentale. Par exemple, toute activité de cuisine débute systématiquement par la sortie des ustensiles, puis celle des ingrédients, etc. Cette façon de faire, habituellement présentée sur support papier à l'aide de pictogrammes a été retranscrite le plus fidèlement possible dans les modèles d'activités de l'expérimentation.

Parmi les recettes possibles, celles concernant les pâtes ont été proposées par les intervenants. Non seulement ces activités ne sont pas complètement inconnues des bénéficiaires, mais il est de plus aisé d'en trouver des variations. Du point de vue expérimental, ce n'est pas une mais au moins deux activités de même charge cognitive qui doivent en effet être réalisées. L'une le sera avec technologie et l'autre sans, pour pouvoir établir des critères de comparaison.

Enfin, les outils d'assistance envisagés ont été présentés aux intervenants du groupe de

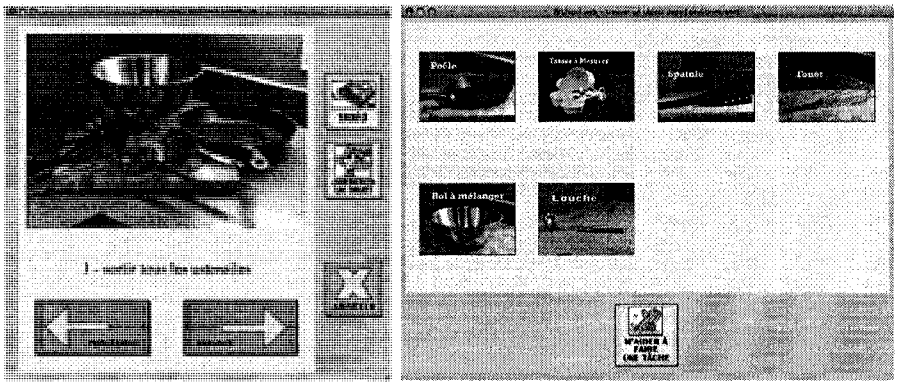

FIGURE 8.6 – Interfaces web des outils d'assistance procédurale (gauche) et d'aide à la localisation des objets (droite)

travail, tout particulièrement l'aide à la localisation des objets et l'assistance procédurale. La mise en oeuvre actuelle de ces actes d'assistance repose en grande partie sur des interfaces *web* (figure 8.6, ainsi que 8.1 à 8.3), il était donc intéressant de les valider auprès des intervenants. L'organisation graphique, limitée à l'essentiel, a été appréciée, l'utilisation sur demande des ressources comme les vidéos explicatives supposée adaptée. Il a par contre été suggéré que les objets à localiser soient liés à l'étape courante uniquement, alors qu'initialement la localisation portait sur un ensemble d'items reliés à l'activité. Enfin, les intervenants ont insisté sur le vocabulaire textuel et visuel utilisé, qui doit rester simple, sans équivoque et cohérent. Cela confirme l'idée selon laquelle un effort conséquent doit être mis au niveau de la représentation de l'activité, tant sur le plan structure et contraintes de réalisation que sur le plan informations d'assistance. Car tout l'effort algorithmique peut être ruiné par un simple message mal formulé. Dans la même lignée, la modélisation actuelle des activités, basée sur les actions de la personne et non sur les objets concernés par ces actions, ne permet pas d'utiliser l'information visuelle associée aux items décrivant ces objets comme support pour l'assistance procédurale. Celle-ci le sera pourtant pour l'aide à la localisation. Autrement dit, une étape reliée à un objet et ce même objet peuvent être illustrés différemment, d'où un risque d'incohérence dans l'information présentée.

8.3.2 Phase d'expérimentation

Protocole expérimental et objectifs

L'expérimentation s'est déroulée en trois temps, soit pour chaque personne impliquée une session de groupe et deux sessions individuelles, conduites par Dany Lussier-Desrochers.

L'objectif de la session de groupe (deux ou trois personnes) est d'initier les participants à la technologie. Dans un premier temps, sont discutées les différences entre une cuisine conventionnelle et la cuisine de l'appartement DOMUS. C'est l'occasion de découvrir les dispositifs technologiques présents dans cet environnement et d'apprendre à manipuler grâce à un jeu de tic-tac-toe l'écran tactile, qui reste l'interface centrale de ce déploiement. Dans un second temps, le chercheur invite les participants à utiliser la technologie pour les aider à réaliser une recette de crêpes. L'interface de l'assistant procédural est expliquée, la relation entre cette interface et le support papier habituellement utilisé lors des cours de cuisine ayant été améliorée grâce au travail réalisé en amont avec les intervenants. On montre aussi comment réagit le système lorsque l'on commet une erreur, en validant l'étape courante alors qu'elle n'est pas finie. L'aide à la localisation des objets est elle aussi illustrée. Cette session se termine autour de la dégustation de quelques crêpes.

Les sessions individuelles portent sur l'évaluation de l'orthèse. Les participants ont été répartis de manière aléatoire selon deux groupes équitables. Le groupe A a réalisé une première recette avec la technologie, puis la deuxième sans l'orthèse, inversement pour le groupe B (tableau 8.1). Les deux activités étaient de même charge cognitive, soit deux variantes d'une recette de pâtes avec une sauce à la viande. Il s'agissait donc d'un devis expérimental équilibré à deux conditions (avec et sans technologie).

TABLEAU 8.1 – Répartition des conditions expérimentales

	Temps 1	Temps 2
Groupe A	avec technologie	sans technologie
Groupe B	sans technologie	avec technologie

Pour chacune de ces activités, le chercheur se tenait à disposition du participant pour répondre à ces questions, et éventuellement intervenir en cas de besoin. Une personne de confiance était aussi présente sur les lieux (dans la plupart des cas l'aidant professionnel), mais restait à l'écart de la réalisation de l'activité. Le participant pouvant néanmoins faire appel à cette personne, notamment s'il souhaitait mettre un terme à l'expérimentation,

conformément au protocole éthique approuvé par les comités des universités impliquées. A partir de là, l'impact de l'orthèse sur l'autonomie fonctionnelle du participant est mesuré par le nombre d'interventions d'assistance offertes par le chercheur afin d'aider la personne à compléter sa recette. Ce calcul est réalisé pour chacune des conditions expérimentales.

Remarques sur les méthodes d'analyse

L'analyse de l'expérimentation est réalisée en deux temps. Le premier a permis d'établir un ensemble de résultats préliminaires qui sont présentés dans cette thèse. Le second est en cours de réalisation, sous la forme d'un projet de Maîtrise en Psychologie mené sous la direction de Dany L.Desrochers. Les résultats de cette seconde étude ne seront donc pas présentés dans ce mémoire.

Le calcul du nombre de comportements d'assistance offerts par le chercheur a été réalisé grâce aux données enregistrées au cours des expérimentations. La librairie domus.experimentation (voir figure 3.1) offre un ensemble d'outils permettant de consigner le déroulement d'une expérimentation. Plus particulièrement, un assistant numérique personnel (PDA) équipé d'une connexion internet sans-fil peut être utilisé par le chercheur pour spécifier à Archipel qu'il vient d'assister le client, *via* une interface *web* présentant les quelques boutons nécessaires. Archipel disposant de sa propre horloge, il est indispensable pour l'analyse que l'estampillage temporel des interventions du chercheur soit réalisé conformément à celle-ci. Dans le cas contraire, il devient délicat de faire le rapprochement entre ces interventions et le comportement de l'application, celui-ci pouvant aussi être pris en consigne. De plus, l'utilisation du PDA ne limite pas la liberté de mouvement du chercheur, et son utilisation reste discrète et *a priori* non perturbante pour le participant. A noter par contre que la prise en consigne des interventions n'a à l'heure actuelle aucune influence sur le comportement d'Archipel en terme d'assistance.

Néanmoins, il est admis que le chercheur puisse commettre des erreurs, en négligeant de prendre en note certaines interventions ou en effectuant des consignes erronées, découlant d'une mauvaise analyse du comportement du participant. Dès lors, ces résultats ne peuvent pas être considérés comme définitifs. Par contre, différentes observations sur l'utilisabilité de l'orthèse et sur quelques considérations techniques ont pu être réalisées au cours des expérimentations et sont présentées dans ce mémoire.

Pour pouvoir effectuer une analyse *a posteriori* des expérimentations, celles-ci ont été filmées. Grâce à ces vidéos (plus de 36 heures de données), il sera non seulement

possible de valider le nombre d'interventions du chercheur mais aussi de les codifier. Cette codification permettra d'associer à chaque intervention une sémantique, notamment relative au niveau d'abstraction de l'intervention. Il existe en effet une différence entre dire à la personne ce qu'elle doit faire, et l'inciter à regarder une vidéo expliquant ce qui doit être réalisé. Dans le second cas, l'autonomie n'est pas favorisée par le chercheur mais par la technologie. L'intervention du chercheur reste, mais selon un niveau élevé d'abstraction. C'est ce travail de codification qui fait l'objet d'un projet de Maîtrise en Psychologie.

Profil d'assistance utilisé pour la condition avec technologie

Assistance procédurale Chaque activité réalisée avec l'orthèse bénéficiait d'une assistance procédurale de niveau intermédiaire. Les étapes présentées correspondaient aux grandes divisions de la recette : sortie des ustensiles, sortie des ingrédients, préparation des légumes, cuisson de la viande et des légumes, cuisson des pâtes, égouttage et service.

Aide à la localisation L'aide à la localisation était offerte pour la sortie des ustensiles et celle des ingrédients, soient les étapes à consonance localisation de l'activité. Le bouton "trouver les objets" de l'interface web (voir figure 8.6) devenait alors accessible, étant grisé dans les autres cas. Les pages d'objets associées étaient limitées aux éléments en lien avec l'étape concernée.

Problèmes de planification La procédure d'assistance suivante était utilisée pour aborder les problèmes de planification :

1. rappel audio de l'étape, interrogeant sur l'effectivité de la réalisation (voir illustration 8.3) ;
2. – invitation à utiliser le localisateur pour les étapes à consonance localisation, l'interface web passant automatiquement de l'assistance procédurale au localisateur d'objets ;
   – lancement de la vidéo explicative pour les autres.

Dans un premier temps, on envisageait d'inclure dans cette procédure un acte d'assistance revenant à modifier le niveau d'abstraction de l'assistance procédurale pour l'étape en cours. D'une assistance procédurale de niveau intermédiaire, on passerait alors à une présentation des tâches concrètes de l'étape. Ainsi, "sortir les ustensiles" deviendrait

l'ensemble d'étapes "sortir les tasses à mesurer" puis "sortir le couteau", *etc.*, celles déjà réalisées étant ignorées. Mais cette idée à été mise de côté avant l'expérimentation, car la présentation de l'information procédurale devenait problématique. Pour ne pas perdre la personne, il faut arriver à lui faire comprendre qu'on lui demande de "sortir les tasses à mesurer" *dans le cadre de* l'étape "sortir les ustensiles". Or aucune solution simple n'a été trouvée pour représenter visuellement cette notion d'inclusion.

Enfin, on notera qu'aucune contrainte temporelle n'avait été spécifiée. S'agissant des problèmes d'attention, la présence d'au moins trois personnes non différenciées dans l'habitat ne permettait pas d'envisager leur gestion.

Perception et compréhension du contexte d'assistance cognitive S'agissant des événements implicites découlant de la manipulation des ustensiles et des ingrédients, il était difficile de disperser ustensiles et ingrédients de manière à ne disposer que d'événements discriminants. En effet, bien que les possibilités de rangement soient nombreuses et toutes équipées d'un contact électromagnétique détectant leur ouverture/fermeture, et pour les plans de travail d'un détecteur de mouvement, les ressources technologiques du côté des effecteurs utilisés pour la localisation des objets ne permettaient l'équipement que d'une douzaine d'emplacements. Dès lors, une validation explicite était nécessaire pour la plupart des étapes. Seuls l'allumage de la plaque de cuisson, associée à la cuisson de la sauce, et l'utilisation de la robinetterie, associée au remplissage de la casserole pour la cuisson des pâtes, étaient considérés comme discriminants. Enfin, quelques étapes, comme la découpe des légumes, ne reposaient que sur une confirmation explicite, aucun dispositif n'étant à même d'en percevoir la réalisation.

Résultats préliminaires

Les figures 8.7 et 8.8 présentent le nombre d'interventions d'assistance offertes par le chercheur, par participant (8.7) et par condition expérimentale (8.8).

Ces résultats préliminaires montrent que l'orthèse aurait un effet sur l'autonomie fonctionnelle des participants. On constate en effet que les participants auraient moins besoin d'assistance de la part du chercheur pour compléter leur recette lorsque l'orthèse est présente. Ces chiffres seront confirmés et affinés lors de l'analyse des enregistrements vidéos. Par contre, on peut dès maintenant présenter un certain nombre d'observations réalisées au cours de ces expérimentations.

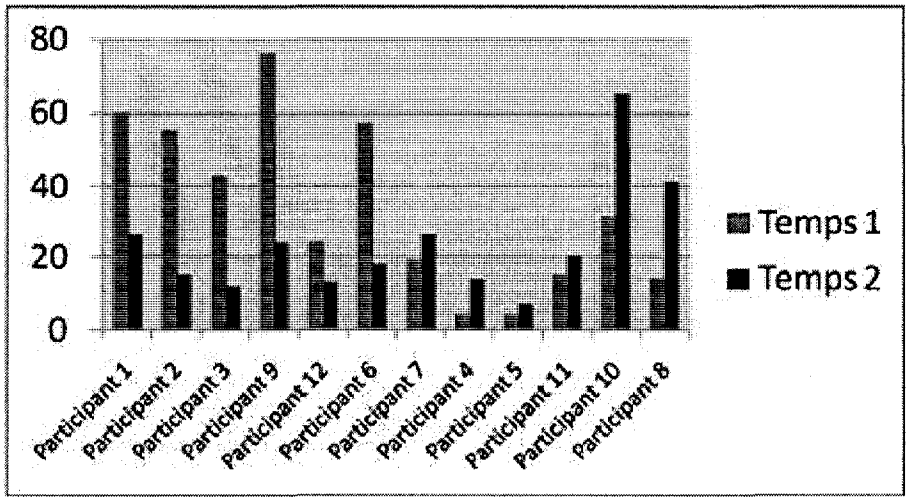

FIGURE 8.7 – Nombre d'interventions par participants pour les deux conditions expérimentales. Les participants 1, 2, 3, 9, 12 et 6 appartiennent au groupe B (sans - avec), les participants 7, 4, 5, 11, 10 et 8 au groupe A (avec - sans)

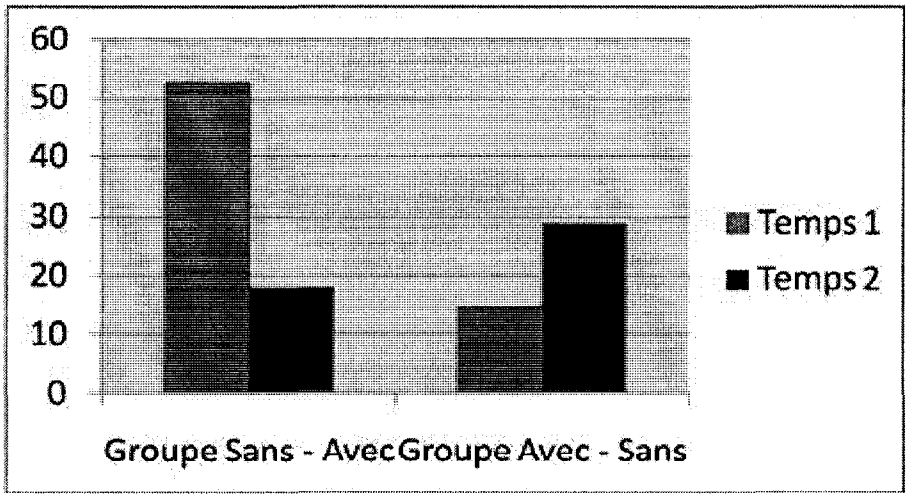

FIGURE 8.8 – Nombre d'interventions par condition expérimentale pour les deux groupes

Utilisabilité de l'orthèse

De manière générale, on observe que l'aide apportée par le chercheur en présence de la technologie est essentiellement liée à l'utilisation de cette dernière, peu d'aide étant donnée spécifiquement pour la réalisation de la recette. Dans la condition sans technologie, l'aide est exclusivement donnée pour cette réalisation. Dès lors, on peut penser qu'un apprentissage plus adéquat de l'orthèse et l'ajustement de son utilisabilité permettront de limiter significativement l'aide en lien avec la technologie et d'améliorer l'autonomie fonctionnelle.

Aide fournie par les interfaces *web* de l'écran tactile (assistance procédurale et aide à la localisation) Ces interfaces sont compréhensibles pour la presque totalité des participants. L'écran tactile simplifie clairement les interactions avec l'orthèse, facilitant d'autant son utilisation. Par contre, quelques participants appuient trop longtemps sur les boutons, produisant l'effet de plusieurs interactions explicites consécutives. Cela a parfois eu pour incidence de confirmer involontairement la réalisation d'une étape qui ne sous-tendait aucune autre forme de validation. Une approche à considérer serait l'ajout de contraintes temporelles spécifiant un temps minimal de réalisation pour chaque étape.

Les photographies destinées à illustrer les étapes sont parfaitement appropriées à la clientèle. Par contre, les séquences vidéos ne sont pas spontanément utilisées par les participants, qui préfèrent demander au chercheur de l'information sur la procédure à utiliser.

L'assistance à la localisation est beaucoup utilisée. Toutefois, son interface nécessiterait quelques améliorations :

- pour les étapes de sortie des ustensiles et des ingrédients, l'illustration est une composition d'images des ustensiles ou des ingrédients à localiser et sortir. Or l'organisation des images au sein de cette composition, réalisée à la main, ne se retrouve pas dans l'interface du localisateur. Le localisateur présente en effet les mêmes images, prises de façon individuelles, mais selon une organisation non prévisible car laissée aux soins du *proxy* de l'écran, à qui l'on demande de diffuser l'aide à la localisation. Cela est source de confusion chez les participants, les deux vues présentant la même information mais selon une organisation potentiellement différente. De plus, la demande d'assistance à la localisation devient effective lorsque l'on appuie sur l'illustration de l'étape. Le scénario suivant s'est donc produit : le participant appuie sur la partie de l'illustration représentant l'objet qu'elle recherche. Comme

l'appui sur l'écran tactile est trop long, l'aide à la localisation est déclenchée pour l'objet dont la représentation est située au même endroit mais dans l'interface du localisateur. L'objet en question ne correspondant pas à celui de l'illustration de l'étape, il y a pour le participant une incohérence entre l'assistance demandée et celle fournie, ce qui est à nouveau source de confusion ;

- quelques participants éprouvent de la difficulté à "sortir" du localisateur, i.e. à revenir à l'assistance procédurale. Pour l'utilisateur, les deux interfaces se présentent comme deux pages *web* entre lesquelles on navigue par un simple bouton[7]. Malgré la présence d'un texte et d'un icône jugés explicites par les intervenants, il est difficile pour certains de passer d'un mode à un autre. Un apprentissage plus important est à considérer ;
- l'utilisation des boutons permettant de naviguer entre les étapes, et donc de confirmer la réalisation de l'étape courante, pose des problèmes pour certains. Avec le bouton "précédent", la personne peut visualiser les étapes déjà réalisées, dont le nom est alors rayé. L'idée est par ce moyen de favoriser la mémoire de travail. Mais la modification visuelle de l'information (nom rayé) ne semble pas explicite. La personne est alors perdue, souvent amenée à appuyer de nombreuses fois sur le bouton "suivant", et à valider des étapes non réalisées (si celles-ci ne sous-tendent aucune autre forme de validation) ;

Aide fournie par les interfaces réparties dans l'environnement Les participants éprouvent une grande difficulté avec l'aide automatiquement fournie par l'orthèse, et qui du point de vue de l'utilisateur démarre seule et sans prévenir. Ainsi, les messages vocaux diffusés suite à une erreur de planification ne produisent pas l'assistance espérée. On peut d'ailleurs supposer que les participants ne portent tout simplement pas attention à ces messages. De même, le fait d'activer automatiquement la diffusion d'une vidéo n'a absolument pas le même impact que lorsque la personne la demande elle-même, même si ce type d'assistance n'a été que peu utilisé.

A côté de cela, le système lumineux servant à mettre en valeur certains emplacements de l'environnement, tout particulièrement pour l'aide à la localisation, est très efficace. Comme précisé dans l'illustration 8.2, un signal sonore avise l'utilisateur de la mise en

[7]En fait, chaque page ne contient aucun lien physique vers l'autre. Le fait de cliquer sur un bouton revient à demander explicitement un type d'assistance à Archipel. Celui-ci répond en demandant aux effecteurs appropriés de fournir l'assistance demandée. Tout cela reste parfaitement transparent pour l'utilisateur, tout en étant modulaire pour le développeur.

oeuvre de cette assistance. Actuellement, un signal sonore précède toute aide non vocale (cela est configurable). L'ensemble permet à la personne de (1) prendre conscience que l'orthèse tente de lui fournir une information, puis (2) chercher cette information, à son temps. On comprend qu'il n'en est rien lorsqu'un message vocal, peut être trop complexe, est diffusé sans avertissement. A noter que les surfaces réfléchissantes (porte blanche du réfrigérateur en l'occurrence), qui reflètent la lumière des diodes électroluminescentes utilisées comme interfaces lumineuses, apportent une certaine confusion chez certains. C'est en effet la source lumineuse qui représente l'information et non pas la surface éclairée par réflection.

Observations techniques

Une situation problématique est intervenue dans de nombreuses expérimentations. L'utilisation de la robinetterie de la cuisine, détectée grâce aux débitmètres installés sur la tuyauterie, est associée de façon discriminante au remplissage de la casserole d'eau pour la cuisson des pâtes. Or de nombreux participants ont souhaité se servir un verre d'eau, rincer les oignons avant de les éplucher ou bien encore se nettoyer les mains après avoir coupé les légumes. Dans tous les cas Archipel s'est manifesté, qualifiant ces actions d'erreur de planification - ce n'était pas le moment de remplir la casserole d'eau - et diffusant un message vocal d'assistance. Pendant les expérimentations, ce comportement a été présenté comme un *bug* de l'orthèse dont il ne fallait pas tenir compte. Grâce à l'interface développeur d'Archipel, il est d'ailleurs possible d'annuler un diagnostic et de continuer le monitorage sans autres conséquences. Dans le cas contraire, le diagnostic maladroit risquerait de bloquer l'assistance d'erreurs réelles. Malheureusement il ne s'agit absolument pas d'un *bug*, mais bien du comportement attendu de l'orthèse, qui n'est par contre pas adapté à la situation. Du point de vue de l'activité telle que décrite, cet événement est en effet discriminant. Mais il est clair que la représentation des activités ne sera jamais parfaite ni exhaustive. On retombe dès lors dans un problème de gestion de l'incertitude, qui n'a été abordé que succinctement jusqu'à présent, avec la notion d'indice de réalisation. Le problème se pose toutefois sous un angle nouveau, puisque les actions réalisées par la personne sont correctes, mais non attendues. La problématique est ouverte.

Au cours des expérimentations réalisées, Archipel ne s'est jamais arrêté de fonctionner. On ne parle pas ici de la justesse de son comportement, telle que discutée précédemment, mais de sa robustesse en tant qu'application informatique. Cela n'a pas par contre pas

été le cas des dispositifs utilisés pour réaliser les interactions homme-machine, tout particulièrement les dispositifs courant-porteur, permettant de contrôler les systèmes lumineux. Or il faut voir l'orthèse comme un tout, l'application Archipel étant dépendante du bon fonctionnement des autres ressources technologiques. Le problème de la tolérance aux fautes devra naturellement être abordé. En l'occurrence, le mauvais fonctionnement des dispositifs lumineux a eu pour conséquence de bloquer les autres actes d'assistance. En effet, ce type d'assistance attend une rétroaction de la part de l'utilisateur. Mais en l'absence de lumière allumée indiquant l'emplacement recherché, il devient difficile de savoir quel tiroir ou placard il faut aller ouvrir, ou sur quel plan de travail il faut chercher. A noter que le problème ne se situait pas au niveau de la communication entre l'application et le modem courant-porteur, assurée *via* ethernet, un tel problème pouvant être détecté au niveau logiciel.

8.4 Conclusion

Ce chapitre a permis d'illustrer le bien fondé des orientations prises pour cette recherche et l'effectivité de sa mise en oeuvre.

Les différents déploiements démontrent tout d'abord la faisabilité et la réalité des développements présentés dans ce mémoire. Pour la première fois sont mis bout à bout les différentes facettes de la sensibilité au contexte d'assistance cognitive : perception, monitorage et utilisation à des fins d'assistance, avec comme tronc commun la représentation des connaissances. Ils renseignent aussi sur les capacités d'adaptation du prototype, dans l'esprit d'ouverture que doit être celui d'un *framework*.

Avec les phases d'expérimentation, on dépasse les considérations techniques pour rentrer dans le vif du sujet : l'orthèse permet-elle d'améliorer l'autonomie fonctionnelle des personnes avec troubles cognitifs ? Grâce aux rencontres réalisés avec les intervenants, le côté client d'Archipel a pu être validé et amélioré. Puis l'expérimentation menée au sein de l'appartement-laboratoire DOMUS a permis de réaliser une première évaluation concrète.

Même si les résultats ne sont que préliminaires, on note que l'orthèse ubiquitaire permet aux participants de réaliser leur recette avec une aide moindre de la part d'un tiers. Par contre, l'aide fournie concerne essentiellement l'utilisation de l'orthèse. Ainsi, on peut conclure que la démarche mise en oeuvre va dans le bon sens, car elle répond aux besoins des utilisateurs. Cependant, la façon dont elle apporte cette réponse doit être

améliorée que pour son utilisabilité soit complète. La mise en oeuvre de certains actes d'assistance, on pense par exemple aux messages vocaux, est à revoir complètement. A côté de cela, d'autres actes ont donné de très bons résultats. Un travail conséquent reste donc à faire, mais cela n'est pas une surprise. Archipel n'est qu'un prototype, mais un prototype dont les fondations sont solides. Et il s'agit de la première étape qui se devait d'être franchie au cours de cette thèse.

Ainsi, du point de vue technique, l'expérimentation a mis en avant différents problèmes ou faiblesses, qui se situent au niveau de la mise en oeuvre des concepts du *framework*. Par exemple, il faut continuer à travailler sur la gestion de l'incertitude dans la mise en oeuvre du monitorage. De même, Archipel a été développé selon certaines restrictions : une unique personne ne réalisant qu'une activité à la fois. L'exemple du verre d'eau pris pendant la préparation des pâtes en montre les limites : cette activité très simple est bien réalisée en parallèle de la recette.

Dans la conclusion générale qui suit, on revient sur les différentes étapes du projet Archipel. Celui-ci a été mené de bout à bout, de la conception à l'expérimentation, avec des résultats qui, en l'état actuel de la recherche sur les orthèses cognitives et malgré l'immense travail qui doit encore être réalisé, sont non seulement intéressants mais aussi novateurs.

Conclusion générale

Dans un laboratoire d'informatique, quelque part au Québec, au cours de la préparation d'une recette de poutine. Dans un coin de la pièce, un intervenant en réadaptation regarde attentivement la scène...

L'ORTHÈSE. Etes-vous certain d'avoir sorti le couteau à fromage en grains ?

LE CHERCHEUR EN INFORMATIQUE. Le quoi ?

L'ORTHÈSE. Une instance de instrument tranchant comportant une lame et un manche dédié à une substance alimentaire obtenue par coagulation du lait, de type cheddar caillé frais.

LE CHERCHEUR EN INFORMATIQUE. Et si tu m'indiquais plutôt où il est ...

On entend un son, puis une lumière s'allume quelque part dans l'environnement

LE CHERCHEUR EN INFORMATIQUE. Ok je l'ai !

L'ORTHÈSE. C'est pas la peine de le dire, je le sais.

LE CHERCHEUR EN INFORMATIQUE. Etape suivante ! Ah oui, les frites. Comment je fais ça ?

Face au chercheur, ce qui ne semblait n'être qu'une surface bien ordinaire s'anime soudain, et une séquence vidéo est diffusée ...

L'ORTHÈSE. Le découpage et la cuisson des frites est une étape cruciale de la préparation de la poutine. Vous devez [...] Voilà comment il faut procéder pour découper les pommes de terre. *fin de la séquence vidéo*

Quelques très longues minutes plus tard ...

L'ORTHÈSE. Vous semblez éprouvez de la difficulté à terminer cette étape.

LE CHERCHEUR EN INFORMATIQUE. Oui, et bien chacun sa spécialité ...

Bien plus tard ...

LE CHERCHEUR EN INFORMATIQUE, *à l'intervenant en réadaptation.* Alors, qu'en pensez-vous ?

L'INTERVENANT. Et bien c'est très intéressant, mais j'ai quelques doutes sur son utilisabilité avec la clientèle ...

LE CHERCHEUR EN INFORMATIQUE. A vrai dire, c'est pour cela que je vous ai convié à cette démonstration. Et si on discutait autour d'une bonne poutine ?

L'INTERVENANT, *regardant le résultat de la recette préparée par le chercheur.* Euh ...

L'ORTHÈSE. Souhaitez-vous que je passe commande chez St-Hubert ?

Dans ce travail de thèse, on s'est efforcé d'établir une relation pertinente entre un ensemble de besoins et une trousse d'outils technologiques, de la formaliser, d'en proposer une mise en oeuvre et de la valider.

Les besoins découlent des troubles cognitifs acquis ou développementaux, qui limitent l'autonomie fonctionnelle et donc la capacité des personnes à réaliser leurs activités de la vie quotidienne (AVQ). Parmi ces troubles, quatre ont été présentés au chapitre 1. Il s'agit des troubles mnésiques, d'attention, et exécutifs d'initiation et de planification, qui ont été évalués comme ayant le plus d'impact sur la réalisation des AVQ.

La réponse à ces besoins se trouve dans la compensation de ces troubles cognitifs. On parle alors d'assistance cognitive. Pour réaliser cette compensation, une trousse d'outils technologiques est à disposition. On la supposera composée d'éléments logiciels et matériels, dans les limites des connaissances théoriques et techniques usuelles. La question est alors de déterminer parmi ces outils quels sont ceux qui sont nécessaires à la constitution d'une orthèse cognitive, i.e. un outil technologique d'assistance cognitive, qui soit adapté à ces besoins. Pour cela, il faut exprimer la relation technologie - assistance cognitive, c'est-à-dire expliquer comment la technologie prend part à l'assistance cognitive.

C'est ce que propose le chapitre 2, en définissant un cadre fonctionnel pour l'étude et la conception des orthèses cognitives. Les capacités d'une orthèse y sont déterminées par sa sensibilité au contexte d'assistance cognitive (CAC). Le CAC spécifie l'ensemble des circonstances dans lequel prend part l'assistance cognitive. La personne à qui est destinée l'assistance, et qui possède un profil cognitif dont découle des besoins, est au premier plan du CAC. Ce contexte dispose aussi d'une dimension dynamique, définie par l'ensemble des événements ayant une incidence sur les besoins en assistance de la personne. Par sensibilité au CAC, on désigne tout d'abord la capacité de l'orthèse à percevoir ces événements. Puis, les perceptions successives des différentes situations contextuelles doivent être mises bout à bout pour comprendre le contexte. Comprendre, c'est avant-

tout détecter les erreurs et problèmes de la personne pour pouvoir y répondre, ce qui se concrétise par l'utilisation du CAC. On arrive alors à la partie visible du comportement de l'orthèse : la production d'actes d'assistance, dont le but est d'aider la personne à réaliser ses AVQ, autrement-dit de compenser ses troubles cognitifs.

Dans ce travail, les AVQ que l'on souhaite plus particulièrement assister sont les activités domiciliaires complexes, dont les recettes de cuisine sont un excellent exemple. Pour être au fait de ces activités, et ainsi percevoir les événements liés à leur réalisation et produire des actes d'assistance au plus près de cette même réalisation, il faut envisager la technologie et plus précisément le rapport entre l'homme et la machine selon une approche ubiquitaire. Ainsi les capteurs, effecteurs et autres dispositifs technologiques pourront être déployés dans l'environnement, là où prend place l'activité, et fournir des services d'assistance efficaces tout en restant en arrière-plan. Les interactions entre l'homme et la machine pourront s'étendre de l'explicite vers l'implicite, l'utilisateur vaquant à ses occupations sans se soucier de l'information qu'il se doit d'apporter aux dispositifs technologiques pour être supporté dans ses réalisations.

Le cadre fonctionnel présente une boucle de contrôle naturelle, la sensibilité allant de la perception à l'utilisation (i.e. la production des actes d'assistance), avec une étape intermédiaire de compréhension. Dès lors, il est sensé d'offrir à toute orthèse répondant aux objectifs de ce travail un cadre applicatif logiciel, ou *framework*. En complément des facettes de la sensibilité au CAC, ce cadre fournira les outils pour décrire, manipuler et raisonner sur les connaissances reliées au CAC, présentés dans le chapitre 4. L'ensemble est organisé autour d'un modèle objet ouvert, facilement adaptable aux exigences d'applications diverses. Celui-ci repose sur deux protocoles à méta-objets et un modèle Java de langage à métaclasses complet. Méta-modélisation et mise en oeuvre de patron de conception y sont possibles. C'est le sous-*framework* archipel.lang.

Le chapitre 5 présente pour sa part le sous-*framework* archipel.hci, dédié aux interactions homme-machine. Plus précisément, c'est la mécanique de perception des événements d'interaction implicites, ainsi que celle de mise en oeuvre de requêtes d'interaction machine vers homme qui sont décrits. Perception et émission utilisent largement la représentation et manipulation des connaissances contextuelles, offrant la couche d'abstraction nécessaire pour pouvoir travailler avec n'importe quel environnement de réalisation d'AVQ.

La perception se limite à la mise en forme d'événements atomiques, correspondant aux actions de la personne dans son environnement. La compréhension doit alors intégrer ces événements atomiques pour suivre la réalisation des activités, en se basant sur une

description *a priori* de celles-ci. Le chapitre 6 précise la nature de ces traitements, sous l'angle du monitorage d'activité. Le monitorage a pour but la détection des erreurs lors de la réalisation d'une activité connue, ainsi que leur qualification. La qualification vise à faire le lien entre une erreur et un trouble cognitif, permettant à l'orthèse de penser l'assistance en fonction des troubles détectés. L'ensemble est réalisé sous la gouverne du sous-*framework* archipel.monitoring.

S'agissant des activités domiciliaires complexes, une proposition de mise en oeuvre du monitorage est proposée, inspirée du système conseiller EpiTalk. Un arbre des tâches synthétise les bonnes façons de réaliser l'activité monitorée, qui sont garanties par la spécification de contraintes de réalisation. La violation de ces contraintes, vue comme une erreur dans la réalisation, est détectée par un traitement des événements d'interaction au sein de la structure, et donne lieu à leur qualification.

C'est alors au tour de l'assistance de rentrer en jeu, décrite au chapitre 7 sous la forme du sous-*framework* archipel.assistance. Celui-ci permet la mise en oeuvre de stratégies d'assistance, donnant lieu aux actes d'assistances : présentation des étapes de réalisation d'une activité, aide à la localisation des objets, émission de messages vocaux, présentation de vidéos illustratives, actions directes sur les artefacts de l'environnement, appel à un tiers, *etc.*. Il offre aussi les outils pour permettre à l'orthèse d'avoir une vision plus macro des erreurs qualifiées, et de réguler la diffusion des actes d'assistance.

L'ensemble est illustré puis présenté de façon concrète au chapitre 8, par l'intermédiaire de trois déploiements du prototype réalisé, le plus imposant étant celui réalisé à l'appartement-laboratoire DOMUS. Cet environnement a ainsi servi de plate-forme expérimentale, afin d'évaluer l'orthèse auprès de douze adultes présentant une déficience intellectuelle légère. Ces personnes ont accepté de venir réaliser deux recettes de cuisine dans cet environnement, selon deux conditions expérimentales : avec et sans l'aide de l'orthèse. Les résultats préliminaires montrent une amélioration de leur autonomie en présence de l'orthèse, le chercheur les accompagnant au cours de ces réalisations étant moins amené à les aider lorsque la technologie est présente. L'expérimentation renseigne aussi sur les limites en terme d'utilisabilité de l'orthèse, ainsi que sur quelques problèmes de fond au niveau de la mise en oeuvre de l'assistance. L'ensemble est favorable en ce qui concerne l'approche développée, un travail conséquent restant à réaliser pour faire de l'orthèse Archipel une solution complète pour l'aide à la réalisation d'activités domiciliaires complexes.

Contributions scientifiques

Conception des orthèses cognitives La première contribution concerne la conception des orthèses. Le cadre proposé permet d'en structurer le fonctionnement. Son comportement vis-à-vis d'une situation contextuelle donnée peut y être explicité, offrant la voie à une analyse et comparaison des orthèses. Pris sous l'angle des besoins de la personne, il devient plus simple d'entrevoir ce qui doit être mis en oeuvre du côté technologique pour répondre à ses besoins.

Représentation et utilisation des connaissances en Java Le modèle proposé au chapitre 4 offre, indépendamment du contexte applicatif, un moyen simple de représenter de l'information en Java. Par rapport au langage hôte est offerte la méta-modélisation, sans modification de la plate-forme Java. La manipulation de la connaissance selon les patrons de conception devient possible, ainsi que des raisonnements simples. En tant que citoyen de première-classe du langage, la connaissance est aisément utilisable dans un contexte programmatique.

Informatique ubiquitaire et sensibilité au contexte Il est proposé une approche de la gestion des interactions implicites dans un contexte d'informatique ubiquitaire basée sur la représentation des capteurs et de leurs services d'interaction. Le cadre applicatif permet la mise en oeuvre de mécanismes d'interprétation bas niveau des données capteurs en actions atomiques. Contrairement aux autres approches présentées dans ce mémoire, cette mise en oeuvre est basée sur la description sémantique des capteurs dans un langage de représentation propre, et non sur l'écriture directe d'éléments de code.

Monitorage et représentation des activités de la vie quotidienne Le principe de monitorage d'activité est posé. Une approche de ce principe inspirée du système conseiller EpiTalk est mise en oeuvre, couplée à un modèle hiérarchique de représentation des AVQ. Une définition algorithmique des erreurs de réalisation à des fins de qualification est posée, la qualification étant vue comme l'étape ultime du monitorage.

Formalisation des actes d'assistance La notion d'acte d'assistance a été posée et une mise en oeuvre proposée. Elle intègre les concepts de profil et de stratégie d'assistance, mettant en avant la relation entre la qualification des erreurs de réalisation (notion de diagnostic) et l'assistance à apporter.

Déploiement et expérimentation d'un prototype d'orthèse cognitive ubiquitaire L'orthèse Archipel a été déployée dans trois environnements et expérimentée auprès de douze adulte présentant une déficience intellectuelle légère. Les déploiements sont à consonance ubiquitaire, avec un ensemble de capteurs et d'effecteurs de natures diverses répartis dans l'environnement. Il s'agit de la première réalisation de ce type.

Perspectives

En tant que prototype, Archipel laisse de nombreuses problématiques ouvertes, offrant tout autant de pistes de travail pour la suite de la recherche et des développements. Dans cette jeune thématique de recherche, chaque élément de réponse pose de nouvelles questions. C'est très certainement ce qui en fait l'intérêt, celle-ci prenant place sur le long terme.

Certaines de ces nouvelles problématiques ont été soulevées dans les différents chapitres de ce mémoire. Pour clore les discussions, quelques-unes sont reprises ici, en s'efforçant d'adopter une vision plus globale des thématiques soulevées. Deux grands points vont dès lors être discutés : le premier concerne la *programmation orientée assistance cognitive*, le second la mise en oeuvre d'interactions homme-machine plus judicieuses. Le tout offre diverses perspectives au présent travail.

Programmation orientée assistance cognitive

Le point de départ de cette réflexion est le travail réalisé du côté de la représentation et de l'utilisation des connaissances. La mécanique objet a été utilisée à maintes places dans le *framework*, pour répondre à un besoin commun : représenter et utiliser le CAC.

Les besoins en terme de programmation se sont précisés, spécialisés. Il ne s'agit plus juste d'écrire un programme, mais bien d'écrire un programme *pour* l'assistance cognitive. Pourquoi ne pas parler alors de *programmation orientée assistance cognitive* (POAC) ? Autour de ce concept, un grand nombre de problématiques peuvent être évoquées. Traditionnellement, on devrait tout d'abord parler d'analyse, avec comme outil le cadre fonctionnel du chapitre 2. Mais c'est à propos de la conception qu'il semble très intéressant de creuser. Les patrons d'interaction proposés par D. Maciuszek [98], notamment cités au chapitre 2, démontrent qu'autour de l'assistance cognitive peuvent être isolées de nombreuses problématiques pour lesquelles des patrons de réponses peuvent être proposés.

Maciuszek s'intéresse à des problèmes de haut niveau, on est donc loin des patrons de conception programmatiques, tels que les patrons GoF [57]. La question est ouverte : que peut-on proposer pour la POAC ?

La qualification des erreurs faites par les personnes au cours de la réalisation de leurs activités peut être vue comme un problème de POAC. Dans ce mémoire, cette question a d'ailleurs été soulevée d'un point de vue algorithmique. L'objectif est avant tout de fournir à l'application des outils pour manipuler les erreurs et faire le lien avec les stratégies d'assistance. Dès lors, une question est de savoir dans quelle mesure la qualification doit être fidèle à la réalité, ou bien s'il est possible de l'accepter de façon plus grossière, si cela est bien suffisant pour guider l'orthèse vers la bonne façon d'aider la personne. Il y a fort à faire de ce côté-ci. On notera les premiers travaux réalisés à ce sujet au laboratoire DOMUS, mais sous l'angle de la modélisation cognitive [49]. Par contre, si la qualification doit servir d'outil d'évaluation de la progression d'une pathologie cognitive, sa nature nécessite peut être d'être affinée. A nouveau le lien qui peut être établi avec les travaux effectués à ce sujet grâce à la modélisation cognitive [137] doit être approfondi.

Cette qualification repose aussi sur la représentation des activités. Ce modèle revêt un caractère particulier, puisqu'il décrit non seulement la structure et les contraintes de l'activité, mais il est aussi porteur des informations d'assistance et est couplé à un modèle sémantique. Ces besoins en représentation se sont manifestés au fur et à mesure des développements, et ont été mis en oeuvre autour d'un modèle structurel réalisé précédemment [13, 15]. Du point de vue de l'analyse, ces besoins ressortent clairement du cadre fonctionnel. S'agissant de la mise en oeuvre, une piste d'évolution déjà discutée est la réorganisation des activités autour des objets, la version actuelle se focalisant sur les actions de la personne pour un environnement donné. Par objet, on entend tout ce qui peut matériellement être impliqué dans la réalisation des activités, et potentiellement manipulé par la personne. Une telle logique de représentation n'est pas nouvelle en soi, une modélisation basée sur les objets ayant servie à des travaux sur la reconnaissance d'activité [120]. L'utilisation des objets, qui sont équipés d'étiquettes d'identification à radio-fréquence, est alors détectée grâce à un gant porté par l'utilisateur, intégrant une antenne à radio-fréquence. L'idée ici est un peu différente, dans la mesure où on ne fait pas d'*a priori* sur la façon dont l'utilisation des objets pourra être détectée. D'un côté, on dispose d'une représentation de l'activité indépendante de tout environnement, dont les tâches terminales font état des objets ou catégories d'objets impliqués, et de la

sémantique de l'action de la personne sur les objets (par exemple, prendre un contenant à liquide). De l'autre, la représentation d'un environnement particulier est fourni. L'objectif est alors de calculer, pour cet environnement, quelles sont les interactions implicites qui vont permettre à l'orthèse de suivre la réalisation de l'activité, grâce à une mécanique de monitorage tel que l'arbre des tâches inspiré d'EpiTalk. A la base de ce calcul on trouvera la description sémantique des capteurs et de leur service d'IHM, ainsi que celle des objets, la logique de traitement donnant peut être lieu à l'esquisse de quelques patrons de conception. Prendre un contenant à liquide pourra tout aussi bien se traduire par l'ouverture du placard contenant les verres, que par un événement d'identification par radio-fréquence d'une tasse. Ainsi, on disposera de modèles d'activités indépendants de tout environnement, qui pourront passer de déploiement en déploiement voire quitter le strict monde d'Archipel.

Faire des objets un élément central de la représentation pourrait aussi avoir un intérêt pour la reconnaissance des situations nécessitant une assistance. Lors de la présentation de l'expérimentation, il a été dit que nombre de participants s'étaient servi un verre d'eau pendant la réalisation de l'activité, déclenchant une assistance non fondée de par l'existence d'une relation discriminante entre l'eau et une composante de l'activité. Concrètement, à ce moment de la réalisation, la personne effectue deux activités en parallèle. Pour les monitorer, il faudrait avoir connaissance de ces deux buts, ce qui n'est pas le cas. On peut entrevoir le problème sous l'oeil de la reconnaissance d'activité : le système identifie la seconde activité et annule l'assistance. La reconnaissance est alors vue comme un outil de réflexion sur ce qui semble, de prime abord, être une erreur. La question est comment mettre en oeuvre une telle reconnaissance ?

La piste de réflexion que l'on propose est de se concentrer avant tout sur les objets sans aller jusqu'à l'activité. En l'occurrence, il s'agit d'eau (objet immatériel) et d'un verre, dont on supposera l'utilisation détectable. D'un point de vue sémantique, le verre est un contenant, l'eau un liquide, une relation peut donc exister entre eux. De plus l'eau étant un liquide *a priori* non dangereux, cette situation non prévue peut être considérée comme non problématique. Il n'y a donc pas reconnaissance d'activité. Bien entendu, tout cela reste profondément subjectif, la notion dans de dangerosité étant intimement liée à la personne aidée. A noter d'ailleurs que si l'utilisation du verre n'est pas détectable, celle de l'eau à un moment inusité peut être relativisée selon le même principe.

En fait, se pose la question de la nécessité de connaître forcément le but de la personne. Oui, s'agissant des activités pour lesquelles un besoin a été exprimé, conformément au

principe du monitorage. Pour les autres, l'essentiel n'est-il pas de s'assurer que cela ne nuit pas à la sécurité de la personne ? Certains verront peut être dans cette approche une simplification du problème, le véritable objectif restant la reconnaissance de toute activité réalisée. Cette vision ne sera pas partagée ici : la notion de service d'assistance machine-homme ne se résume pas à un problème théorique de reconnaissance de plan.

Pour terminer sur la POAC, deux thématiques se doivent d'être citées. La première est la répartition des traitements, intimement liée à la logique ubiquitaire. Des travaux sur les architectures multi-agents sont en cours au laboratoire DOMUS, et on rappellera que le système EpiTalk a originellement été pensé dans cette optique. S'agissant du modèle objet de représentation et d'utilisation des connaissances, la question de sa répartition doit être soulevée. La deuxième thématique concerne le déploiement et la gestion d'Archipel. La plate-forme OSGI pourrait être utilisée à cette fin, conformément aux travaux réalisés à DOMUS [61].

Vers des interactions homme-machine plus judicieuses

La seconde grande perspective de ce travail est l'amélioration des interactions entre l'homme et la machine, tout particulièrement en ce qui concerne les actes d'assistance. L'expérimentation a montré qu'il était difficile de préjuger de leur utilisabilité. Il va donc falloir repenser leur conception, pourquoi pas selon une approche centrée utilisateur, puis en réaliser l'évaluation afin de créer des profils d'assistance, en lien avec les profils des clients. Cette dernière notion doit elle-même être définie, divers travaux ayant été réalisés à ce sujet dans des domaines applicatifs connexes [76]. L'amélioration passera peut être par l'utilisation de nouveaux médias, comme les interfaces tangibles, dont une première exploration pour l'assistance cognitive à été réalisée à DOMUS [26].

Une autre piste tout aussi complexe est la mise en place d'une vision plus macro de l'assistance. L'exemple donné dans ce mémoire est celui d'une personne allumant par erreur la plaque de cuisson. Conformément à la supposée séquence en cours, l'orthèse - dans sa version actuelle - s'efforcera de lui indiquer la "bonne" prochaine étape à réaliser, sans se soucier de la malencontreuse modification de l'environnement. Ainsi, c'est l'extinction de la plaque de cuisson qui devrait avoir priorité, très certainement parce que celle-ci représente un danger potentiel. A nouveau la nature des objets impliqués semble peser dans la réflexion.

Ces considérations sur les suites possibles à donner à ce travail en termine la présentation.

Conclusion générale

Nombreux sont les défis qu'il reste à relever pour faire de la technologie un outil valable pour la compensation des troubles cognitifs. Mais la porte est ouverte ...

Annexe A

Schéma XML pour la spécification d'items (archipel.lang)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.domus.usherbrooke.ca"
  elementFormDefault="qualified" attributeFormDefault="qualified"
  xmlns:xdo="http://www.domus.usherbrooke.ca"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="itemCollection">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="includeItemCollection"
          maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:attribute name="itemCollectionId"
              type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="item" maxOccurs="unbounded"
          minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="itemDescription"
                minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="itemProperty"
                      maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element
                            name="propertyName" type="xs:string" />
                          <xs:element
                            name="propertyValue" type="xs:string" />
                        </xs:sequence>
```

```
                    <xs:attribute
                      name="instanceProperty" type="xs:boolean" use="optional"
                      default="true">
                    </xs:attribute>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="itemId" type="xs:string"
          use="required" />
        <xs:attribute name="itemName" type="xs:string"
          use="optional" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="itemCollectionId" type="xs:string"
    use="required" />
  <xs:attribute name="itemCollectionHidden" type="xs:boolean"
    use="optional" />
 </xs:complexType>
</xs:element>
</xs:schema>
```

FIGURE A.1 – Schéma XML pour la spécification d'items

Annexe B

Exemple de document XML de spécification d'items (archipel.lang)

```
<?xml version="1.0" encoding="UTF-8"?>
<xdo:itemCollection xdo:itemCollectionHidden="true"
  xdo:itemCollectionId="contexte materiel"
  xmlns:xdo="http://www.domus.usherbrooke.ca"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.domus.usherbrooke.ca
      http://www.domus.usherbrooke.ca/schemas/archipel_lang.xsd ">
  <xdo:item xdo:itemId="aat:300194567" xdo:itemName="contenant pour boisson">
    <xdo:itemDescription>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>itemClass</xdo:propertyName>
        <xdo:propertyValue/>
      </xdo:itemProperty>
    </xdo:itemDescription>
  </xdo:item>
  <xdo:item xdo:itemId="aat:300043202" xdo:itemName="tasse">
    <xdo:itemDescription>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>decoration</xdo:propertyName>
        <xdo:propertyValue/>
      </xdo:itemProperty>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>extends</xdo:propertyName>
        <xdo:propertyValue>aat:300194567</xdo:propertyValue>
      </xdo:itemProperty>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>itemClass</xdo:propertyName>
        <xdo:propertyValue/>
      </xdo:itemProperty>
    </xdo:itemDescription>
  </xdo:item>
  <xdo:item xdo:itemId="aat:300043202#BLEUE" xdo:itemName="tasse bleue">
```

```
    <xdo:itemDescription>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>decoration</xdo:propertyName>
        <xdo:propertyValue>couleur bleue</xdo:propertyValue>
      </xdo:itemProperty>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>instanceof</xdo:propertyName>
        <xdo:propertyValue>aat:300043202</xdo:propertyValue>
      </xdo:itemProperty>
    </xdo:itemDescription>
  </xdo:item>
  <xdo:item xdo:itemId="aat:300043202#ROUGE" xdo:itemName="tasse rouge">
    <xdo:itemDescription>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>decoration</xdo:propertyName>
        <xdo:propertyValue>couleur rouge</xdo:propertyValue>
      </xdo:itemProperty>
      <xdo:itemProperty xdo:instanceProperty="true">
        <xdo:propertyName>instanceof</xdo:propertyName>
        <xdo:propertyValue>aat:300043202</xdo:propertyValue>
      </xdo:itemProperty>
    </xdo:itemDescription>
  </xdo:item>
</xdo:itemCollection>
```

FIGURE B.1 – Exemple de document XML de spécification d'items

Annexe C

De l'identification des items (archipel.lang)

Associer à chaque item un identifiant lorsque le domaine d'application est l'habitat ou les objets de la vie quotidienne peut rapidement se transformer en casse-tête. Il suffit pour cela de penser aux différentes sortent d'assiettes ou de couteaux susceptibles de résider dans une cuisine, chacun ayant un rôle particulier dans la préparation ou la prise des repas. Du point de vue de l'assistance, ces petites différences seront peut être importantes. Autrement dit, il n'y a pas de raison *a priori* d'en simplifier la représentation.

La solution la plus sage est de laisser cette charge aux spécialistes de la linguistique. Ainsi, le *Art & Architecture Thesaurus (AAT)* de la fondation Getty [148] offre une vaste base de connaissances sur les concepts précités. Rappelons que dans un thesaurus le *vocabulaire* est *contrôlé* : chaque élément possède une définition non-ambigüe et non redondante, dictée par une *autorité de compétence.* Un vocabulaire contrôlé permet d'échanger de l'information tout en s'assurant que le sens sera perçu de la même façon par tous les intervenants, point de départ fort intéressant pour la représentation de connaissances.

Le concept d'autorité de compétence est repris pour l'identification des items sous la forme d'espaces de nommage, dans l'esprit des *namespaces* XML [154]. L'espace de nommage "aat" fera ainsi référence au Thesaurus Getty. Pour chaque item associé à cet espace, il doit être possible d'en vérifier la définition auprès de cette autorité, identifiée par un URI[1]. L'URI pour "aat" sera "www.getty.edu/research/conducting_research/vocabularies/aat/",

[1]Uniform Resource Identifier

soit l'URL[2] de ce thesaurus consultable en ligne[3]. Pour les concepts propres au domaine de l'assistance cognitive, l'espace de nommage du laboratoire DOMUS, baptisé "xdo", sera utilisé. Au final, le patron proposé pour l'identification des concepts est **nms** :**id**, où **nms** est l'espace de nommage et **id** l'identifiant du concept dans cet espace. Si la notion d'instance est présente dans le modèle, il est proposé de compléter le patron par #**instanceId**, la sémantique d'**instanceId** restant locale à une représentation. Néanmoins, on précisera que le rôle de l'identifiant se limite à la définition du concept (i.e. au contrôle du vocabulaire). Cette chaîne de caractère n'a pas pour essence de témoigner de relation entre concepts.

[2]Uniform Resource Locator

[3]Le Thesaurus Getty n'est qu'anglophone. Mais rares sont les ressources de cette nature et richesse utilisables à des fins de recherche.

Annexe D

Framework applicatif domus.xml

Le *framework* applicatif domus.xml facilite l'analyse syntaxique et l'écriture de documents XML dont le contenu dispose en Java d'une représentation objet de même structure. Basé sur SAX pour l'analyse et DOM pour l'écriture, le *framework* ne demande à l'utilisateur que d'associer à chaque balise la classe d'analyse/écriture correspondante, et ce de façon récursive jusqu'aux balises représentant les feuilles du document ou schéma XML. Le *framework* fournit les classes d'analyse/écriture pour les attributs, les types simples, les types complexes indexés (séquence) ou non.

La mécanique repose sur l'introspection des classes utilisées pour la modélisation de la connaissance côté Java. S'il s'agit d'effectuer l'analyse syntaxique d'un document, alors à chaque classe d'analyse doit être associée la classe Java à instancier (passée en argument du constructeur). Lorsqu'une balise ouvrante est rencontrée par l'analyseur (conformément à la spécification SAX), trois situations peuvent se produire :

1. aucune classe d'analyse n'a été associée à cette balise, elle est ignorée ;
2. la classe d'analyse associée témoigne du type complexe de la balise, l'analyseur passe récursivement la main à l'objet chargé de l'analyse de cette sous-structure du document, qui instancie la classe Java associée ;
3. le type analysé est simple ou c'est un attribut, l'objet chargé de l'analyse réalise alors :
   (a) l'invocation dynamique de la méthode valueOf(String) : Object de la classe Java correspondante, créant ainsi la représentation objet de la chaîne de caractères analysée. La méthode valueOf est une méthode de classe prenant comme paramètre une chaîne de caractères et retournant une instance de la

classe concernée initialisée à partir de cette chaîne. Les classes de l'API Java représentant les types de bases (chaînes de caractères, entiers, booléens, *etc.*) disposent toutes d'une méthode avec cette signature. Elle doit être créée pour les classes utilisateurs correspondant à des types simples ou attributs XML. C'est le cas avec l'API domus.time d'Archipel, qui offre des classes compatibles pour la représentation et la manipulation (calcul) des concepts de date, heure, plage horaire, durée, déviation, calendrier (persistance réalisée selon la spécification iCalendar, grâce à l'API tiers iCal4J[1]), horloge et minuteur ;

(b) l'affectation de la valeur dans l'objet père, porté par l'analyseur père, la méthode adéquate étant découverte par introspection à partir du nom de la balise XML.

Le retour de récursion est déclenché lorsqu'une balise fermante connue est recontrée, l'objet analyseur effectuant alors l'affectation de l'objet manipulé, et ainsi de suite. Un exemple est donné ci-dessous pour l'analyse des documents XML représentant les collections d'items (voir annexes A et B pour le schéma correspondant et un exemple de document, ainsi que la figure 4.4 pour le diagramme des classes Java). Dans cet exemple, c'est une instance existante d'ItemCollection qui est "chargée" à partir de la description XML. Il existe l'équivalent pour la création à l'exécution de l'objet racine cible.

```
import ca.usherbrooke.domus.archipel.lang.xml.core.*;
import ca.usherbrooke.domus.xml.handler.*;

/**
 * Methode utilitaire pour l'analyse syntaxique d'une collection d'items.
 * @param fileUrl l'URL du document XML a parser.
 * @param logger l'objet utilise pour tracer l'avancement de l'analyse et ses possibles
     exceptions.
 * @param collection l'instance qui va etre "charge" des items decrits en XML.
 * @return true if no exception has been thrown during the parsing, false otherwise.
 */
public static boolean parseItemCollection(URL fileUrl, ItemCollection itemCollection,
    Logger logger) {

  RootHandler handler = new RootHandler(fileUrl, itemCollection, new
      DefaultHandlerExceptionManager(logger)) {
    /** root level : itemCollection */
    protected void fillHandlerTable() {
      addTagHandler("itemCollection", this);
      addTagHandler("itemCollectionId", new AttributeHandler(String.class));
```

[1]http ://ical4j.sourceforge.net/

```
    addTagHandler("itemCollectionHidden", new AttributeHandler(Boolean.class));
    addTagHandler("includeItemCollection",
      new ComplexIndexedTypeIntrospectorHandler(IncludeItemCollection.class) {
        /** set of included collections */
        protected void fillHandlerTable() {
          addTagHandler("itemCollectionId", new AttributeHandler(String.class));
        }
    });
    addTagHandler("item",
      new ComplexIndexedTypeIntrospectorHandler(Item.class) {
        /** set of items */
        protected void fillHandlerTable() {
          addTagHandler("itemId", new AttributeHandler(String.class));
          addTagHandler("itemName", new AttributeHandler(String.class));
          addTagHandler("itemDescription",
            new ComplexTypeIntrospectorHandler(ItemPropertiesHandler.class) {
              /** item's description */
              protected void fillHandlerTable() {
                addTagHandler("itemProperty",
                  new ComplexIndexedTypeIntrospectorHandler(ItemProperty.class) {
                    /** set of properties, i.e. the item's description */
                    protected void fillHandlerTable() {
                      addTagHandler("instanceProperty", new
                          AttributeHandler(Boolean.class));
                      addTagHandler("propertyName", new
                          SimpleTypeHandler(String.class));
                      addTagHandler("propertyValue", new
                          SimpleTypeHandler(String.class));
                    }
                  });
              }
          });
        }
      });
    }
  };
  /** starts the parsing */
  return handler.parse();
}
```

FIGURE D.1 – Exemple de classe d'analyse syntaxique basée sur le *framework* applicatif domus.xml

L'écriture fonctionne selon un principe similaire. L'utilisateur doit associer à chaque balise qu'il souhaite écrire une classe d'écriture, celle-ci se chargeant par introspection de récupérer les objets à écrire. La méthode toString() : String est utilisée pour les types simples et les attributs comme support de base pour la persistance (soit la représentation de l'objet sous la forme d'une chaîne de caractères). L'ensemble nécessite que les classes

utilisateurs suivent les principes des JavaBeans[2] (respect des patrons de nommage ou constitution d'une classe de description (*BeanInfo*), constructeur public sans paramètres), la présence d'un constructeur unique avec paramètres pouvant toutefois être gérée.

Au final, la mise en oeuvre de ce *framework* applicatif ne nécessite que peu d'effort de la part de l'utilisateur, à partir du moment ou la logique objet des classes d'analyse et d'écriture est admise. Le code est concis (notamment dans l'exemple D.1 de par l'utilisation de classes d'analyses anonymes) et structurellement proche du modèle objet. Cette approche de l'analyse et de l'écriture représente un intérêt si l'utilisateur veut garder la main sur le modèle objet **et** sur la représentation XML, différentes API offrant la création des classes à partir d'une représentation XML ou la gestion de la persistance en XML de classes existantes. C'est le cas pour la représentation des connaissances dans Archipel (items - voir chapitre 4 - et tâches - voir chapitre 6).

[2]http ://java.sun.com/javase/technologies/desktop/javabeans/docs/spec.html

Annexe E

Framework applicatif archipel.fwk.appmanager

Le *framework* applicatif archipel.fwk.appmanager permet de gérer le cycle de vie d'une application, à partir d'une description de la configuration de cette dernière en XML. C'est tout particulièrement intéressant pour Archipel, où les composants et ressources, nombreux et d'origines diverses, doivent être manipulés selon une certaine logique, la configuration en tant que telle devant être ajustée en fonction de l'environnement de déploiement. Le *framework* offre ainsi la modularité nécessaire aux différentes configurations Archipel. En somme, il n'existe pas une application Archipel, mais un vaste ensemble de briques que ce *framework* permet d'assembler à sa guise, ou presque.

La figure E.2 présente une vue partielle des classes du *framework*, tandis que la figure E.1 offre un exemple commenté de document XML de configuration. Voici une explication générale de ces concepts, les noms des classes Java servant de points de repère :

AppConfiguration Une configuration d'application est un ensemble d'informations décrivant l'application. Plus précisément, on retrouve :

AppResource Un ensemble de *ressources*. Par ressource, on entend tout ce dont l'application a besoin pour fonctionner et qui doit chargé au lancement de l'application. Il peut s'agir de modèles de données (les collections d'items pour Archipel), de moteurs divers (par exemple, le moteur d'IHM d'Archipel), *etc.* A toute ressource, dans la description XML, doit être associée le nom d'une classe Java implémentant l'interface AppComponent. C'est par exemple le cas de la classe ItemManager, qui permet de gérer une collection d'items. Peut aussi être associée à la ressource le chemin sur disque d'un fichier (un do-

cument XML, un fichier de propriétés, *etc.*), ainsi que le nom de la classe pouvant servir de vue (dans le sens du patron de conception observateur [57]) implémentant alors AppComponentView. Enfin, l'ordre de manipulation des ressources peut être lui aussi précisé, ce qui est indispensable à la satisfaction des nombreuses dépendances pouvant intervenir entre ressources, comme c'est le cas dans Archipel ;

AppContextParam Des *paramètres contextuels*, i.e. un ensemble couples propriété-valeur liés au déploiement de l'application dans un contexte particulier. Cela reprend le principe des fichiers de propriétés traditionnels et dans une version plus moderne celui des fichiers de déploiement d'applications web (fameux web.xml) ;

AppModeDescription Le but ultime de toute application est de proposer à l'utilisateur un ensemble de fonctionnalités. C'est le principe des *modes* décrits ici. Un mode est l'instance d'une classe implémentant l'interface AppMode, pouvant être initialisée, activée et désactivée. Pour Archipel, l'édition des connaissances est un mode, le monitorage d'activité un autre, *etc.* Bien entendu, les modes ont accès aux ressources de l'application ainsi qu'aux paramètres contextuels ;

AppExecutionContextDescription Archipel, côté développeur ou intervenant (en opposition au côté client), reste une bonne vieille application avec son interface graphique utilisateur. Néanmoins, libre à l'utilisateur de définir à sa façon cet environnement graphique, en précisant le *contexte d'exécution*. Dans le pire des cas, ce contexte pourrait être un pseudo-terminal fonctionnant en ligne de commandes, mais un système de fenêtrage avec un menu déroulant proposant les modes de l'application et les vues des ressources est tout aussi convenable. Le choix est laissé à l'utilisateur, la classe qu'il définira devant implémenter l'interface AppGUIExecutionContext, dévouée à la gestion des modes et vues. A noter que pour toutes les interfaces citées, le *framework* offre des implémentations par défaut diverses.

AppLoader Le *chargeur d'application* est l'objet auquel est dévolu la gestion du cycle de vie de l'application. En fait, le chargeur n'est pas limité qu'au simple chargement. En effet, celui-ci opère selon un mode qui lui a été attribué à un moment donné, chaque mode n'étant qu'une stratégie (patron de conception éponyme) de manipulation des composants de l'application. La mise en oeuvre d'un mode requiert tou-

tefois l'implémentation d'un certain nombre de méthodes (classe AppLoaderMode) ; dès lors, le rôle du chargeur est, pour un mode donné, d'appeler les différentes méthodes de celui-ci. En somme, le chargeur est écrit selon un patron *template* [57]. A l'heure actuelle, 4 modes sont décrits, assurant le chargement, le rechargement, l'arrêt et la sauvegarde de l'application, d'où la présence des méthodes load(...), reload(...), stop(...) et save(...) dans l'interface AppComponent. A noter que des filtres sur les composants peuvent être utilisés pour les différents modes si nécessaire.

Concrètement, Archipel ne dispose d'aucune classe principale, le fameux "main" Java. Le seul "main" utile est celui de ce *framework*. Dans sa version graphique, le mode par défaut étant celui du chargement, le *framework* invite l'utilisateur à spécifier le document XML de configuration, puis se charge du reste, une petite mécanique événementielle permettant de suivre l'évolution du chargement. L'ensemble donne un outil indispensable pour la gestion d'une application complexe comme Archipel.

```
<?xml version="1.0" encoding="UTF-8"?>
<xdo:applicationConfiguration
  xmlns:xdo="http://intranet.domus.usherbrooke.ca"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://domus.usherbrooke.ca
      http://domus.usherbrooke.ca/schemas/applicationConfiguration.xsd">
  <firstResourceToLoad>resource1</firstResourceToLoad>
  <firstResourceToSave>resource2</firstResourceToSave>
  <xdo:applicationResource xdo:resourceId="resource1"
    xdo:resourcePathRelative="true">
    <resourcePath>
      if the resource uses external data, the (relative="true"?) path of the data source
          (.xml, .properties, etc.)
    </resourcePath>
    <resourceClass>
      the class name of this resource, must implement
          archipel.fwk.appmanager.component.AppComponent
    </resourceClass>
    <resourceViewClass>
      if any, the class name of the view of this resource, must implement
          archipel.fwk.appmanager.component.AppComponentView
    </resourceViewClass>
    <nextResourceToLoad>resource2</nextResourceToLoad>
    <nextResourceToSave />
  </xdo:applicationResource>
  <xdo:applicationResource xdo:resourceId="resource2">
    <!-- another resource, that has to be loaded after resource1, but saved before it
        ... -->
    <nextResourceToLoad />
    <nextResourceToSave>resource1</nextResourceToSave>
```

```
  </xdo:applicationResource>
  <!-- others resources ... -->
  <xdo:applicationParam>
    <paramName>name of the parameter</paramName>
    <paramValue>value of the parameter</paramValue>
  </xdo:applicationParam>
  <!-- others paramaters ... -->
  <xdo:applicationMode>
    <modeClass>
      the class name of this mode, must implement archipel.fwk.appmanager.mode.AppMode
    </modeClass>
  </xdo:applicationMode>
  <!-- others modes ... -->
  <xdo:applicationExecutionContext>
    <executionContextManagerClass>
      the class name of the execution context manager, must implement
          archipel.fwk.appmanager.executioncontext.AppExecutionContextManager
    </executionContextManagerClass>
  </xdo:applicationExecutionContext>
</xdo:applicationConfiguration>
```

FIGURE E.1 – Exemple de document XML de configuration

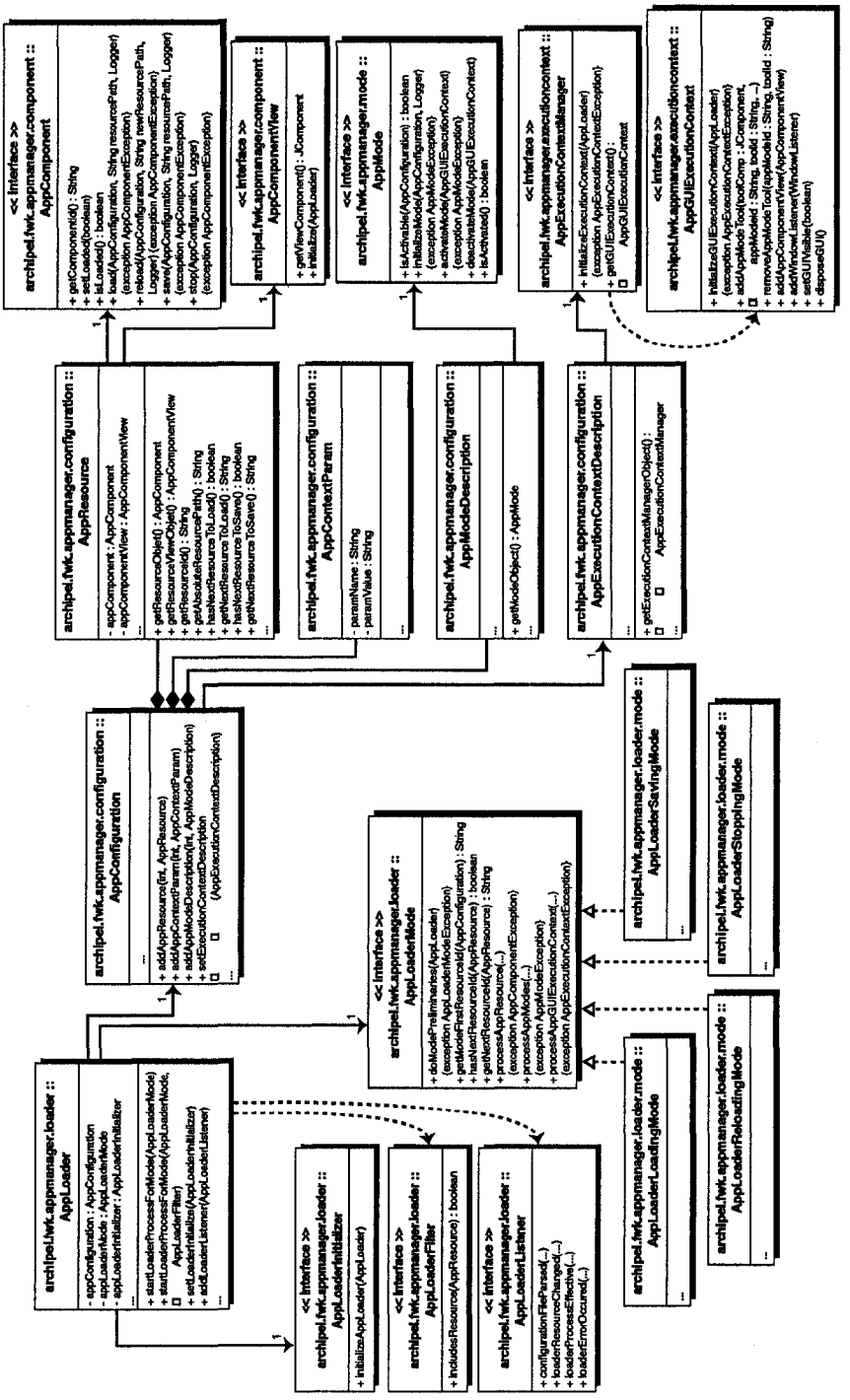

FIGURE E.2 – Diagramme de classes du *framework* de gestion d'application (vue partielle)

Annexe F

Edition graphique des connaissances et *framework* domus.beanbuilder

Le *framework* archipel.lang offre un support adaptable pour la représentation et la manipulation des connaissances (voir chapitre 4). Grâce à son MOP, c'est la sémantique d'un langage de représentation et d'utilisation de la connaissance qui peut être définie, et a fortiori un ensemble de connaissances basées sur le ce langage.

Dans cette annexe sont présentés les outils permettant la manipulation graphique de cette connaissance, pour les modèles structurels et sémantiques. L'objectif est d'offrir à l'utilisateur la transparence des modèles sous-jacents (ne pas avoir à manipuler directement le XML) ainsi qu'une manipulation dont la logique est conforme à la sémantique du langage.

F.1 Collection d'items et feuille de propriétés

La figure F.1 présente une copie d'écran de l'éditeur. La partie gauche offre une vue de la collection d'objets issus du modèle structurel, la partie droite les propriétés de l'objet (item) sélectionné.

Pour la collection d'items, l'éditeur construit la représentation autour d'un composant de type JTree, issu de l'API graphique Java Swing[1]. L'agencement des objets représentant les items au sein de ce composant dépend de la sémantique donnée au langage (voir chapitre 4, tableau 4.2). Dans la figure F.1, l'agencement est traditionnel pour un langage à

[1]http ://java.sun.com/docs/books/tutorial/uiswing/

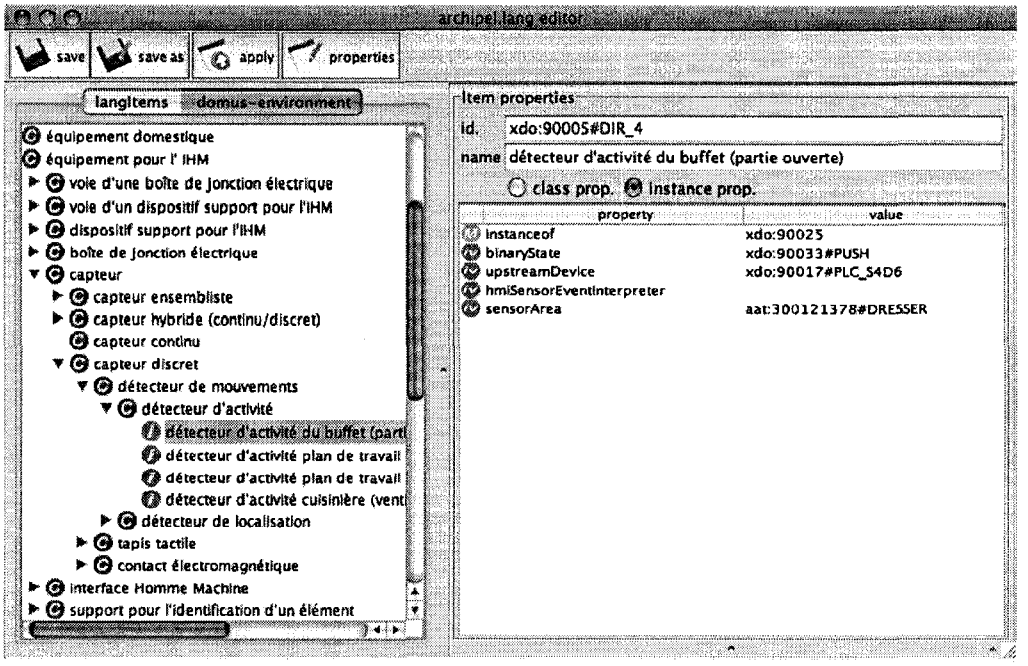

FIGURE F.1 – Edition graphique des connaissances : collection d'items et feuille de propriétés

classes, avec une présentation hiérarchique des hiérarchies de classes, et les items instances comme feuilles.

Du côté des propriétés des items, l'édition utilise une feuille de propriétés, notion généralement associée aux composants Java (Java Beans). Le principe est simple : l'éditeur se charge de découvrir les propriétés de l'objet édité et de les présenter *via* une structure adéquate, en l'occurrence un composant de type JTable dont les éléments sont personnalisés en fonction de la propriété. C'est le *framework* applicatif domus.beanbuilder qui est chargé de ce travail. Le nom de ce *framework* rappelle celui du projet "Bean Builder", héritier des outils BeanBox initialement proposés pour les composants Java par Sun[2]. Domus.beanbuilder reprend les principes de la BeanBox et offre pour n'importe quel objet un support pour l'édition basé sur les capacités d'introspection Java. Il permet de plus de détourner ce principe pour l'appliquer aux items du modèle objet structurel, offrant ainsi une vue naturelle des connaissances décrites.

[2]https ://bean-builder.dev.java.net/

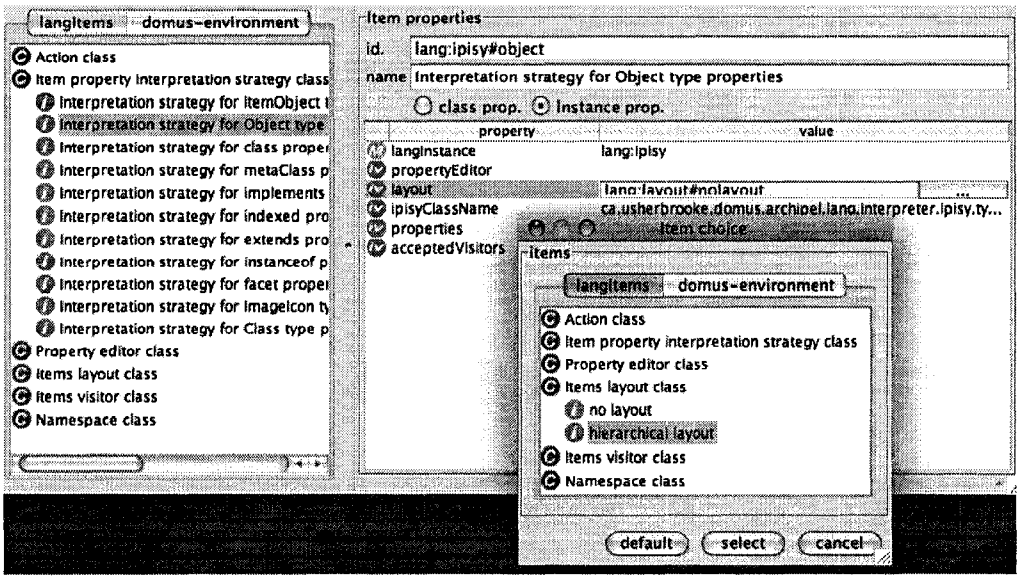

FIGURE F.2 – Edition graphique des connaissances : édition des propriétés

F.2 Edition des propriétés

En effet, il faut se rappeler que l'introspection des objets du modèle structurel ne donnerait absolument pas le résultat présenté. Pour ce faire, c'est dans la sémantique du langage que le *framework* trouve les informations habituellement présentes dans les classes de description des composants Java (BeanInfo). Tout particulièrement, comme le stipule à nouveau le tableau 4.2, il est possible de préciser de quelle façon une propriété doit être éditée. La figure F.2 illustre le cas d'une propriété *layout* (agencement) d'un item représentant une stratégie d'interprétation. On se rappellera en effet que la sémantique du langage est elle aussi décrite selon le même principe et donc manipulable graphiquement dans les même conditions. Un *layout* étant aussi un item, l'éditeur de propriété se concrétise par une boîte de dialogue donnant accès aux collections d'items chargées[3].

F.3 Ajout de connaissances

[3]Et non cachées, ce qui devrait logiquement être le cas de la collection d'items déterminant la sémantique du langage.

L'ajout de connaissances doit elle aussi suivre la logique du langage de représentation. Deux possibilités peuvent intervenir : (1) la création *ex nihilo* de connaissance, ou (2) l'ajout d'une connaissance liée à l'existant. Dans un contexte de langage à classes, le premier cas correspondrait à la création d'une classe racine d'un arbre d'héritage[4], le second à celui d'une classe héritant d'une existante ou tout simplement à l'ajout d'une instance (l'essence précédant l'existence).

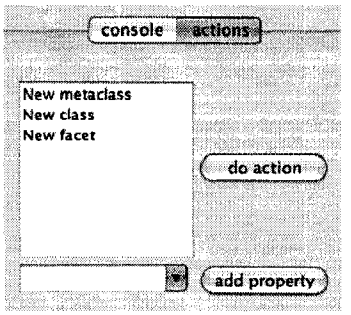

FIGURE F.3 – Edition graphique des connaissances : ajout de connaissances *ex nihilo*

Du point de vue graphique, ces deux possibilités sont respectivement illustrées par les figures F.3 et F.4, la nature de la manipulation offerte découlant des spécifications du langage, sous les concepts d'*actions* et de *visiteurs*. Quelque soit la façon de créer de la connaissance, l'idée à appliquer est la même : (1) créer un nouvel item, instance de la classe Item dans le modèle structurel, (2) ajouter des propriétés à cet item, et (3) interpréter ces propriétés pour pouvoir insérer l'item à sa juste place dans la vue et en permettre la manipulation. Dès lors, décrire une action ou un visiteur revient sur le fond à préciser la ou les propriétés à ajouter à l'item créé. Sur la forme, les actions sont liées au langage, alors que les visiteurs sont associés aux stratégies d'interprétation des propriétés. Concrètement, les actions sont présentées sous la forme d'une simple liste (figure F.3). Pour les visiteurs, il s'agit d'un menu contextuel dont le contenu est fonction de l'item sélectionné. Ce contenu correspond en quelque sorte aux visiteurs que l'item accepte de recevoir, tel que déterminé par ses propriétés. Il y a dans cette technique un peu du patron de conception "visiteur" [57][5].

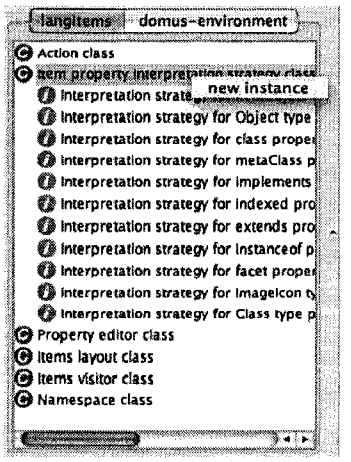

FIGURE F.4 – Edition graphique des connaissances : ajout de connaissances à partir de l'existant

[4]Du point de vue du modèle sémantique, la racine de toute hiérarchie d'héritage est la classe ItemObject. Mais cette notion n'apparaît pas telle quelle dans le modèle structurel.

[5]En fait, la composition graphique JTree + JTable, utilisé pour représenter les données d'un côté et les éditer de l'autre, est définie au niveau du *framework* domus.beanbuilder, et intègre par défaut une

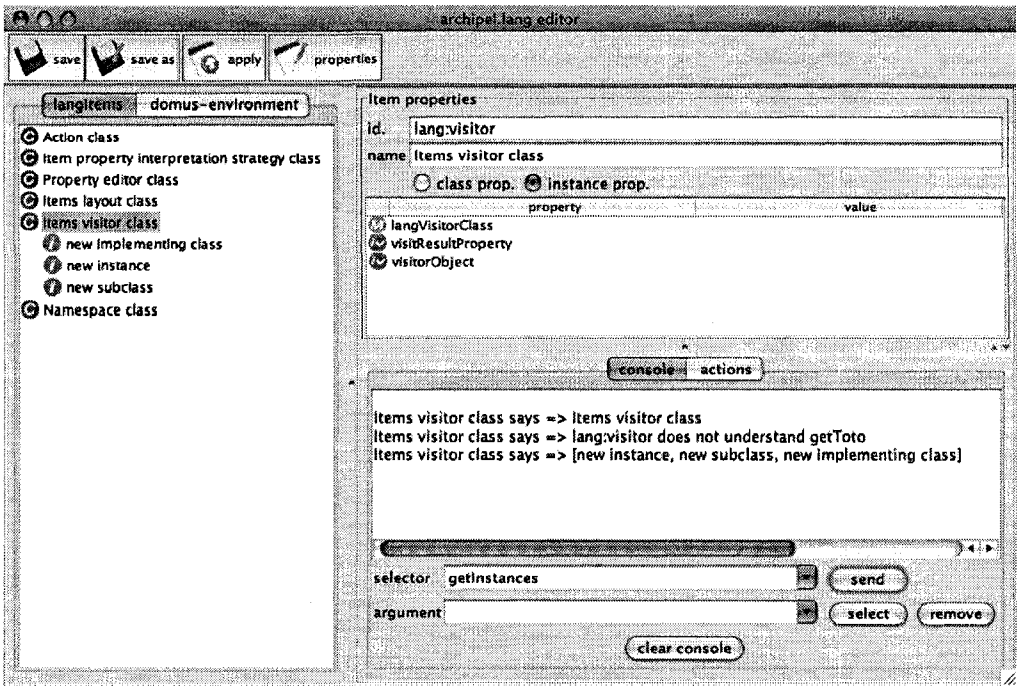

FIGURE F.5 – Edition graphique des connaissances : console d'envoi de message

F.4 Utilisation du modèle objet sémantique

Une fois la connaissance créée, l'étape ultime est la mise en oeuvre du modèle objet sémantique. Si ce modèle a été chargé pour une collection d'items, il devient intéressant de pouvoir interagir avec cette connaissance par envoi de messages. C'est ce que permet la console présentée à la figure F.5. En l'occurrence, le message "getInstances" a été envoyé à la classe des visiteurs, l'objet sémantique correspondant ayant répondu en conséquence[6]. Un message non compris se traduira par un "does not understand" à la console[7]. A noter que s'agissant des arguments des messages, seuls les chaînes de caractères et les itemObjets peuvent être passés.

Au final, les outils présentés permettent d'éditer les connaissances tout en suivant la sémantique du langage de représentation spécifiée par l'utilisateur, en rendant transparentes les techniques de représentation sous-jacentes (XML, modèle structurel). L'en-

mécanique de visiteur qui est reprise en partie ici.

[6]Par défaut, la collection d'items présentant la sémantique du langage est automatiquement chargée sous sa forme sémantique. L'accès aux informations contenues (actions, stratégies d'interprétation, visiteurs, agencements et éditeurs de propriétés) n'est réalisé que par envoi de messages.

[7]Les messages sont ici réifiés, pour pouvoir notamment prendre note des exceptions soulevées.

semble fait d'archipel.lang un *framework* complet, ou presque, pour la représentation, l'édition et l'utilisation de connaissances en Java.

Annexe F : Edition graphique des connaissances et framework domus.beanbuilder

Bibliographie

[1] « Jena - A Semantic Web Framework for Java ». http ://jena.sourceforge.net/.

[2] G. D. ABOWD, A. BOBICK, I. ESSA, E. MYNATT et W. ROGERS. « The Aware Home : Developing Technologies for Successful Aging ». Dans *AAAI Workshop and Automation as a Care Giver*, 2002.

[3] J. AHOLA. « Ambient Intelligence ». *European Research Consortium for Informatics and Mathematics (ERCIM) News*, 47, 2001.

[4] J. L. AMBITE et C. KNOBLOCK. « Planning by rewriting ». *Journal of Artificial Intelligence Research*, pages 207–261, 2001.

[5] F. ARMENGAUD. *La pragmatique.* Presses Universitaires de France, 2007.

[6] J. BACHEVALIER. « *Les troubles de mémoire et les système de mémoire* », Chapitre 23, pages 391–413. Neuropsychologie clinique et neurologie du comportement. Presses Universitaires de Montréal, 2005.

[7] A. BADDELEY. *Working Memory.* Oxford University Press, 1986.

[8] R. BAECKER. « Designing Electronic Memory Aids : A Research Framework ». Dans *Workshop on Designing for People with Cognitive Impairments, CHI 2006*, 2006.

[9] R. BAECKER. « A Taxonomy of Technology for Cognition ». Dans *International Conference on Technology and Aging (ICTA)*, 2007. (affiche).

[10] J. BARDRAM. « The Java Context-Awareness Framework (JCAF) - A service infrastructure and programming framework for context-aware applications ». Dans *Pervasive 2004*, pages 98–115, 2004.

[11] J. E. BARDRAM. « Applications of Context-Aware Computing in Hospital Work - Examples and Design Principles ». Dans *SAC'04*, March 14-17 2004.

[12] J. E. BARDRAM et T. R. HANSEN. « The AWARE Architecture : Supporting Context-Mediated Social Awareness in Mobile Cooperation ». Dans *CSCW'04*, pages 192–201, November 6-10 2004.

[13] J. BAUCHET. « Une approche de la reconnaissance d'activité pour l'assistance dans les habitats intelligents ». Mémoire de maîtrise, Université de Sherbrooke, (QC) Canada, novembre 2005.

[14] J. BAUCHET, D. LUSSIER-DESROCHERS, H. PIGOT et S. GIROUX. « A pervasive assistance to foster people with cognitive disabilities autonomy », 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc). September 11, 2007. Montreal, Canada 2007. (démonstration).

[15] J. BAUCHET et A. MAYERS. « A Modelisation of ADLs in its Environment for Cognitive Assistance ». Dans Sylvain GIROUX et Hélène PIGOT, éditeurs, *3rd International Conference On Smart Homes and health Telematic (ICOST)*, Assistive Technology Research Series, pages 221–228. IOS Press, July. 4-6 2005.

[16] J. BAUCHET, H. PIGOT et S. GIROUX. « Cognitive assistance based on ubiquitous computing, mobile computing and cognitive modeling ». Dans *Smart House Workshop. International Conference on Aging, Disability and Independance (ICADI)*, February 1 2006.

[17] J. BAUCHET, D. VERGNES, S. GIROUX, H. PIGOT et J. P. SAVARY. « Assisting Cognitively Impaired People in Smart Homes : A Pervasive Prototype Applied to Morning Routine ». *The Computer Journal*, Soumis.

[18] J. BAUCHET, D. VERGNES, Sylvain GIROUX et Hélène PIGOT. « A Pervasive Cognitive Assitant For Smart Homes ». Dans *International Conference on Aging, Disability and Independance*, page 228, February 1-5 2006.

[19] M. BEAUDOUIN-LAFON. « Interfaces Homme-Machine : vue d'ensemble et perspectives ». *Génie Logiciel & Systèmes experts*, 24 :4–16, 1991.

[20] Y. BELLIK. « *Interfaces multimodales : concepts, modèles et architectures* ». Thèse de doctorat, Université Paris XI, France, 1995.

[21] M. BLOUIN. « IIDRIS (Index International et Dictionnaire de la Réadaptation et de l'Intégration Sociale) ». http ://www.med.univ-rennes1.fr/iidris/.

[22] J. BOGER, J. HOEY, P. POUPART, C. BOUTILIER, G. R. FERNIE et A. MIHAILIDIS. « A Planning System Based on Markov Decision Processes to Guide People with Dementia Through Activites of Daily Living ». *IEEE Transactions on Information Technology in BioMedicine*, 10(2) :323–333, 2006.

[23] T. BOTEZ-MARQUARD et F. BOLLER. *Neuropsychologie clinique et neurologie du comportement.* Presses Universitaires de Montréal, 2005.

[24] B. BOUCHARD. « *Un modèle de reconnaissance de plan pour les personnes atteintes de la maladie d'Alzheimer basé sur la théorie des treillis et sur un modèle d'action en logique de description* ». Thèse de doctorat, Université de Sherbrooke, (QC) Canada, 2006.

[25] B. BOUCHARD, A. BOUZOUANE et S. GIROUX. « A Keyhole Plan Recognition Model for Alzheimer's Patients : First Results ». *Journal of Applied Artificial Intelligence (AAI), Accepted for publication in volume 22, (7)*, pages 1–34, July 2007.

[26] B. BOUSSEMART et S. GIROUX. « Tangible User Interfaces for Cognitive Assistance ». Dans *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, 2007.

[27] P. BRETON et S. PROULX. *L'explosion de la communication : introduction aux théories et aux pratiques de la communication.* La Découverte, Paris, 2006.

[28] O. BUCUR, P. BEAUNE et O. BOISSIER. « Defining and Modeling Context in a Multi-Agent Systems Architecture for Decision-Making ». Dans *International Workshop on Managing and Reasoning on Context (MRC 05), in conjunction with IJCAI 2005*, 2005.

[29] S. CARMIEN. « End User Programming and Context Responsiveness in Handheld Prompting Systems for Persons with Cognitive Disabilities and Caregivers ». Dans *Conference on Human Factors in Computing Systems CHI'05*, Extended Abstracts on Human Factors in Computing Systems, pages 1252–1255, April 2-7 2005.

[30] S. CARMIEN, M. DAWE, G. FISCHER, A. GORMAN, A. KINTSCH et J. F. jr SULLIVAN. « Socio-technical environments supporting people with cognitive disabilities using public transportation ». *ACM Trans.Comput.-Hum.Interact.*, 12(2) :233–262, 2005.

[31] T. CHAARI, F. LAFOREST et A. CELANTANO. « Service-Oriented Context-Aware Application Design ». Dans *First International Workshop on Managing Context Information in Mobile and Pervasive Environments (MCMP)*, 2005.

[32] D. CHAMBERLAND-TREMBLAY, J. BAUCHET, H. PIGOT et S. GIROUX. « La médiation par l'espace intelligent, collaboration et communauté de soins à l'intérieur du domicile ». Dans *Recontres en Intelligence Collective*, mai 2006.

[33] P. COINTE. « Metaclasses are First Class : the ObjVLisp Model ». Dans *Conference proceedings on Object-oriented programming systems, languages and applications (OOPSLA)*, pages 156–162, 1987.

[34] D. COLBRY, B. PEINTNER et M. E. POLLACK. « Execution Monitoring with Quantitative Temporal Bayesian Networks ». Dans *6th International Conference on AI Planning and Scheduling*, 2002.

[35] E. COLE. « Cognitive prosthetics ; an overview to a method of treatment ». *NeuroRehabilitation*, 12 :39–51, 1999.

[36] E. COLE et M. K. Jr MATTHEWS. « *Cognitive Prosthetics and Telerehabilitation : Approaches for the Rehabilitation of Mild Brain Injuries* », pages 111–120. HSW-Distorsion & Leichte Traumatische Hirnverletzung Behanlungskonzepte. 2001.

[37] R. COOPER et T. SHALLICE. « Contention scheduling and the control of routine activities ». *Cognitive Neuropsychology*, 17(4) :225–232, 2000.

[38] P. COSTANZA et R. HIRSCHFELD. « Language Constructs for Context-oriented Programming - An Overview of ContextL ». Dans *Dynamic Languages Symposium, OOPSLA'05*, October 18 2005.

[39] J. COUTAZ, L. BALME, C. LACHENAL et N. BARRALON. « Software Infrastructure for Distributed Migratable User Interfaces ». Dans *Workshop At the Crossroads : The Interaction of HCI and Systems Issues in Ubicomp'03.*, 2003.

[40] J. COUTAZ et G. REY. « Foundations for a theory of Contextors ». Dans J. VANDERDONCKT et C. KOLSKI, éditeurs, *Computer-Aided Design of User Interfaces III*, pages 13–34. Kluver Academic, 2002.

[41] A. Van DAM. « User Interfaces : Disappearing, Dissolving, and Evolving ». *Communications of the ACM*, 44(3) :50–52, 2001.

[42] D. K. DAVIES et M. L. WEHMEYER. « Enhancing Independent Task Performance for Individuals with Mental Retardation Through Use ot a Handheld Self-Directed Visual and Audio Prompting System ». *Education and Training in Mental Retardation and Developmental Disabilities*, 37(2) :209–217, 2002.

[43] R. DAVIS, H. SHROBE et P. SZOLOVITS. « What is Knowledge Representation ? ». *AI Magazine*, 14(1) :17–33, 1993.

[44] A. K. DEY. « *Providing Architectural Support for Building Context-Aware Applications* ». Thèse de doctorat, Georgia Institute of Technology, USA, 2000.

[45] A. K. DEY et G. D. ABOWD. « Towards a Better Understanding of Context and Context-Awareness ». Dans *Workshop on The What, Who, Where, When and How of Context-Awareness, 2000 Conference in Human Factors in Computing Systems (CHI 2000)*, 2000.

[46] A. K. DEY, G. D. ABOWD et D. SALBER. « A Context-Based Infrastructure for Smart Environments ». Dans *1st International Workshop on Managing Interactions in Smart Environments*, pages 114–128, 1999.

[47] A. K. DEY, M. FUTAKAWA, D. SALBER et G. D. ABOWD. « The Conference Assistant : Combining Context-Awareness with Wearable Computing ». Dans *3rd International Symposium on Wearable Computers (ISWC '99)*, pages 21–28, October 20-21 1999.

[48] A. K. DEY, D. SALBER et G. D. ABOWD. « A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications ». *Human-Computer Interaction*, 16 (2-4) :97–166, 2001.

[49] A. DION et H. PIGOT. « Modeling cognitive errors in the realization of an activity of the everyday life ». Dans *Cognitio 2007*, June 15 - 17 2007.

[50] E. DISHMAN. « Inventing Wellness Systems for Aging in Place ». *Computer*, 37(5) :34–41, 2004.

[51] C. DONY, J. MALENFANT et D. BARDOU. « *Les langages à prototypes* », Chapitre 8. Langages et modèles à objets. INRIA, 1998.

[52] C. DUYCKAERTS et J. J. HAUW. « *Introduction à l'anatomie fonctionnelle et à la pathologie du système nerveux humain* », Chapitre 2, pages 23–58. Neuropsychologie clinique et neurologie du comportement. Presses Universitaires de Montréal, 2005.

[53] M. FITZMAURISE, H. ISHII et W. BUXTON. « Bricks : Laying the Foundations for Graspable User Interfaces ». Dans *CHI'95*, pages 442–449, 1995.

[54] C. FLOERKEMEIER et F. SIEGEMUND. « Improving the Effectiveness of Medical Treatment with Pervasive Computing Technologies ». Dans *Workshop on Ubiquitous Computing for Pervasive Healthcare Applications at Ubicomp'03*, 2003.

[55] T. FLURY. « *Modélisation et gestion de l'information de localisation physique pour la composition de dispositifs d'interface humaine dans les espaces intelligents* ». Thèse de doctorat, Université Grenoble I - Joseph Fourier, France, 2004.

[56] Stanford Center for BIOMEDICAL INFORMATICS RESEARCH. « The Protégé Ontology Editor and Knowledge Acquisition System ». http ://protege.stanford.edu/.

[57] E. GAMMA, R. HELM, R. JOHNSON et J. VLISSIDES. *Design patterns : elements of reusable object-oriented software.* Addison-Wesley Professional, 1995.

[58] M. GHALLAB, D. NAU et P. TRAVERSO. *Automated Planning : theory and practice.* Morgan Kaufmann, 2004.

[59] M. GOLM et J. KLEINÖDER. « metaXa and the Future of Reflection ». Dans *OOPSLA Workshop on Reflective Programming in C++ and Java*, 1998.

[60] P. GORMAN, R. DAYLE, C. A HOOD et L. RIMRELL. « Effectiveness of the ISAAC cognitive prosthetic system for improving rehabilitation outcomes with neurofunctional impairment ». *NeuroRehabilitation*, 18 :57–67, 2003.

[61] C. GOUIN-VALLERAND et S. GIROUX. « Managing and deployment of applications with OSGi in the context of Smart Homes ». Dans *3rd IEEE international conference on Wireless and Mobile Computing, Networking and Communications 2007 (Wimob 07)*. IEEE, 8-10 octobre 2007.

[62] P. GRAF et B. UTTL. « Prospective memory : A new focus for research. ». *Consciousness and Cognition : An International Journal*, 10(4) :437–450, 2001.

[63] R. GRIMM, J. DAVIS, E. LEMAR, A. MACBETH, S. SWANSON, S. GRIBBLE, T. ANDERSON, B. BERSHAD, G. BORRIELLO et D. WETHERALL. « Programming for pervasive computing environments ». Rapport Technique UW-CSE-01-06-01, University of Washington, Department of Computer Science and Engineering, 2001.

[64] T. GROSS. « Ambient Interfaces : Design Challenges and Recommendations ». Dans *10th International Conference on Human-Computer Interaction*, pages 68–72, 2003.

[65] Information Society Technologies Advisory GROUP. « Scenarios for Ambiant Intelligence for 2010 ». Rapport Technique, European Commission, Community Research, 2001. ftp ://ftp.cordis.europa.eu/pub/ist/docs/istagscenarios2010.pdf.

[66] A. HELAL, H. YANG, J. KING et B. ROSE. « Atlas - Architecture for Sensor Network Based Intelligent Environments ». *ACM Transactions on Sensor Networks (Submitted)*, 2007.

[67] N. HERSH et L. TREADGOLD. « Neuropage : the rehabilitation of memory dysfunction by prosthetic memory and cueing ». *Neurorehabilitation*, 4 :465–486, 1994.

[68] T. T. HEWETT, R. BAECKER, S. CARD, T. CAREY, J. GASEN, M. MANTEI, G. PERLMAN, G. STRONG et W. VERPLANK. « Curricula for Human-Computer Interaction ». Rapport Technique, ACM Special Interest Group on Computer-Human Interaction, 1992.

[69] A. HORGAS et G. D. ABOWD. « *The Impact of Technology on Living Environments for Older Adults* », Chapitre 9, pages 230–252. Technology for Adaptive Aging. The National Academies Press, 2004.

[70] R. HÉBERT, J. DESROSIERS, N. DUBUC, M. TOUSIGNANT, J. GUILBEAULT et E. PINSONNAULT. « Le système de mesure de l'autonomie fonctionnelle (SMAF) ». *La Revue de Gériatrie*, 28(4) :323–336, 2003.

[71] Taligent INC.. « Building Object-Oriented Frameworks, A Taligent White Paper ».

[72] E. A. INGLIS, A. SZYMKOWIAK, P. GREGOR, A. F. NEWELL, N. HINE, P. SHAH, B. A. WILSON et J. J. EVANS. « Issues surrounding the user-centred developement of na new interactive memory aid ». *Universal Access in the Information Society*, 2(3) :226–234, 2003.

[73] Organisation internationale de normalisation ISO. « Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11 : Guidance on usability », 1998. standard ISO 9241-11.

[74] H. ISHII et B. ULLMER. « Tangible Bits : Towards Seamless Interfaces between People, Bits and Atoms ». Dans *CHI'97*, 1997.

[75] E. JANSEN, B. ABDULRAZAK, H. I. YANG, J. KING et S. HELAL. « A Programming Model for Pervasive Spaces ». Dans *Third International Conference on Service-Oriented Computing (ICSOC 2005)*, LNCS 3826, 2005.

[76] R. KADOUCHE. « *Modélisation du profil utilisateur et personnalisation dans les espaces de vie intelligents* ». Thèse de doctorat, Institut National des Télécommunications, Paris, France, 2007.

[77] N. KARA et A. DRAGOI. « Reasoning with contextual data in telehealth applications ». Dans *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*, 2007.

[78] S. KATZ, A. B. FORD, R. W. MOSKOWITZ, B. A. JACKSON et M. W. JAFFE. « Studies of Illness in the Aged. The Index of ADL : a Standardized Measure of Biological and Psychological Function ». *JAMA : the Journal of the American Medical Association*, 185 :914–919, Sep 21 1963.

[79] H. KAUTZ, O. ETZIONI, D. FOX et D. WELD. « Foundations of Assisted Cognition Systems ». Rapport Technique CSE-02-AC-01, University of Washington, 2003.

[80] R. KEAYS et A. RAKOTONIRAINY. « Context-Oriented Programming ». Dans *Third International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'03)*, September 19 2003.

[81] G. KICZALES, J. M. ASHLEY, L. RODRIGUEZ, A. VAHDAT et D. G. BOBROW. « *Metaobjects protocols : Why we want them and what else they can do* », pages 101–118. Object-Oriented Programming : The CLOS Perspective. MIT Press, Cambridge, MA, 1993.

[82] G. KICZALES, J. des RIVIÈRES et D. G. BOBROW. *The Art of the Metaobject Protocol.* The MIT Press, Cambridge, MA, 1991.

[83] H. J. KIM, D. T. BURKE, M. M. DOWDS, K. A. Robinson BOONE et G. J. PARK. « Electronic memory aids for outpatient brain injury : follow-up findings ». *Brain Injury*, 14(2) :187–196, 2000.

[84] H. J. KIM, D. T. BURKE, M. M. DOWDS et J. GEORGES. « Utility of a microcomputer as an external memory aid for a memory-impaired head injury patient during in-patient rehabilitation ». *Brain Injury*, 13(2) :147–150, 1999.

[85] I. KORHONEN et J. E. BARDRAM. « Guest Editorial Introduction to the Special Section on Pervasive Healthcare ». Dans *IEEE Transactions on Information Technology in Biomedicine*, 2004.

[86] I. KORHONEN, P. PAAVILAINEN et A. SÄRELÄ. « Application of ubiquitous computing technologies for support of independent living of the elderly in real life settings ». Dans *UbiHealth 2003 : The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*, 2003.

[87] Y. LACHAPELLE et M. L. WEHMEYER. « *L'autodétermination* », pages 203–214. Manuel professionnel sur la déficience intellectuelle. Gaëtan Morin, Montréal, Canada, 2003.

[88] F. LATFI, B. LEFEBVRE et C. DESCHENEAUX. « Ontology-Based Management of the Telehealth Smart Home, Dedicated to Elderly in Loss of Cognitive Autonomy ». Dans *OWL : Experiences and Directions. Third International Workshop (OWLED 2007)*, 2007.

[89] M. P. LAWTON et E. M. BRODY. « Assessment of older people : self-maintaining and instrumental activities of daily living ». *The Gerontologist*, 9(3) :179–186, 1969.

[90] R. LEVINSON. « The Planning and Execution Assistant and Trainer (PEAT) ». *Journal of Head Trauma Rehabilitation*, 12(2) :85–91, 1997.

[91] R. LEVINSON. « A Custon-Fitting Cognitive Orthotic That Provides Automatic Planning and Cueing Assistance ». Dans *Technology and Persons with Disabilitites Conference*, 2004.

[92] M. D. LEZAK. *Neuropsychological Assesment.* Oxford University Press, 1995.

[93] E. J. Van LOENEN. « On the role of Graspable Objects in the Ambient Intelligence Paradigm ». Dans *Smart Objects Conference SOC'2003*, 2003.

[94] E. LOPRESTI, A. MIHAILIDIS et N. KIRSCH. « Assistive technology for cognitive rehabilitation : State of the art ». *Neuropsychological Rehabilitation*, 14(1/2) :5–39, 2004.

[95] D. LUSSIER-DESROCHERS, Y. LACHAPELLE, H. PIGOT et J. BAUCHET. « Apartments for People with Intellectual Disability : Promoting Innovative Community Living Services ». Dans *2nd International Conference on Intellectual Disabilities/-Mental Retardation.*, November 6-8 2007.

[96] D. LUSSIER-DESROCHERS, Y. LACHAPELLE, H. PIGOT et J. BAUCHET. « Des habitats intelligents pour promouvoir l'autodétermination et l'inclusion sociale ». *Revue Francophone de Déficience Intellectuelle*, 18, 2007.

[97] K. LYYTINEN et Y. YOO. « Issues and Challenges in Ubiquitous Computing ». *Communications of the ACM*, 45(12) :62–65, 2002.

[98] D. MACIUSZEK, J. ABERG et N. SHAHMEHRI. « What help do older people need ? : constructing a functional design space of electronic assistive technology applications ». Dans *Assets '05 : Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pages 4–11. ACM Press, 2005.

[99] P. MAES. « Concepts and Experiments in Computational Reflection ». Dans *ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'87)*, pages 147–155, 1987.

[100] G. MASINI, A. NAPOLI, D. CONET, D. LÉONARD et K. TOMBRE. *Les langages à objets : langages de classes, langages de frames, langages d'acteurs.* InterEditions, Paris, 1989.

[101] C. E. MCCARTHY et M. E. POLLACK. « A Plan-Based Personnalized Cognitive Orthotic ». Dans *6th International Conference on AI Planning and Scheduling*, 2002.

[102] T. MEULEMANS, F. COLLETTE et M. VAN DER LINDEN. *Neuropsychologie des fonctions exécutives.* Solal, 2004.

[103] A. MIHAILIDIS, J. C. BARBENEL et G. R. FERNIE. « The efficacy of an intelligent cognitive orthosis to facilitate handwashing by persons with moderate-to-severe dementia ». *Neuropsychological Rehabilitation*, 14(1) :135–172, 2004.

[104] A. MIHAILIDIS, B. CARMICHAEL, J. BOGER et G. FERNIE. « An Intelligent Environment to Support Aging-in-Place, Safety, and Independence of Older Adults with Dementia ». Dans *UbiHealth 2003 : The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*, 2003.

[105] A. MIHAILIDIS et G. R. FERNIE. « Context-aware assistive devices for older adults with dementia ». *Gerontechnology*, 2(2) :173–188, 2003.

[106] A. NAPOLI, B. CARRÉ, R. DUCOURNAU, J. EUZENAT et F. RECHENMANN. « *Objets et représentation, un couple en devenir* », volume 10 de *Des octets aux modèles. Vingt ans après : où en sont les objets ?*, pages 61–81. RSTI - L'objet, 2004.

[107] J. NIELSEN. *Usability Engineering.* Academic Press, Boston, MA, 1993.

[108] M. E. O'CONNELL, C. A. MATEER et K. A. KERNS. « Prosthetic systems for addressing problems with initiation : guidelines for selection, training, and measuring efficacy ». *NeuroRehabilitation*, 18(1) :9–20, 2003.

[109] Congress of the United States of AMERICA. « Technology-Related Assistance for Individuals With Disabilities Act of 1988 », 1988. Public Law 100-407.

[110] F. PACHET. « Langages à objets et représentation des connaissances, Mémoire d'habilitation, Université Paris 6, France », 1997.

[111] F. PACHET et S. GIROUX. « Building plan recognition systems on arbitrary applications : the spying technique ». Dans *IJCAI'95 Workshop on "New generation of plan recognition systems"*, 1995.

[112] F. PACHET, S. GIROUX et G. PAQUETTE. « Pluggable Advisors as Epiphyte Systems ». Dans *Computer Aided Learning in Science and Engineering (Calisce '94)*, 1994.

[113] R. A. PAGADALA et S. A. NAPPER. « An Adaptive Cognitive Orthosis Shell for Task Guidance ». Dans *Sixteenth Southern Biomedical Engineering Conference*, pages 422–424. IEEE Press, 1997.

[114] G. PAQUETTE, F. PACHET, S. GIROUX et J. GIRARD. « Epitalk, generating advisor agents for existing information systems ». *Artificial Intelligence in Education*, 7(3-4) :349–379, 1996.

[115] J. PARADISE, E. D. MYNATT, C. WILLIAMS et J. GOLDTHWAITE. « Designing a cognitive aid for the home : a case-study approach ». *SIGACCESS Access.Comput.*, 77-78 :140–146, 2004.

[116] D. PATTERSON, Lin LIAO, Dieter FOX et Henry KAUTZ. « Inferring High-Level Behavior from Low-Level Sensors ». Dans *UbiComp'03*, 2003.

[117] D. J. PATTERSON, O. ETZIONI, D. FOX et H. KAUTZ. « Intelligent Ubiquitous Computing to Support Alzheimer's Patients : Enabling the Cognitively Disabled ». Dans *UbiComp'02*, 2002.

[118] D. J. PATTERSON, L. LIAO, K. GAJOS, M. COLLIER, N. LIVIC, K. OLSON, S. WANG, D. FOX et H. KAUTZ. « Opportunity Knocks : A System to Provide Cognitive Assistance with Transportation Services ». Dans *UbiComp'04*, volume 3205 de *LNCS*, pages 433–450. Davies, N., 2004.

[119] J. F. PERROT. « *Objets, classes, héritage : définitions* », pages 3–31. Langages et modèles à objets. INRIA, 1998.

[120] M. PHILIPOSE, K. P. FISHKIN, M. PERKOWITZ, D. J. PATTERSON, D. HAHNEL, D. FOX et H. KAUTZ. « Inferring Activities from Interactions with Objects ». *IEEE Pervasive Computing*, 3-4 :50–57, 2004.

[121] H. PIGOT, J. BAUCHET et S. GIROUX. « *Assistive devices for people with cognitive impairments* », Chapitre 12. The Engineering Handbook on Smart Technology for Aging, Disability and Independence. John Wiley & Sons, 2007.

[122] H. PIGOT, D. LUSSIER-DESROCHERS, J. BAUCHET, Y. LACHAPELLE et S. GIROUX. « A smart home to assist recipes' completion. ». Dans *Festival of International Conferences on Caregiving, Disability, Aging and Technology (FICCDAT), 2nd International Conference on Technology and Aging (ICTA)*, June 16-19 2007.

[123] H. PIGOT, D. LUSSIER-DESROCHERS, J. BAUCHET, Y. LACHAPELLE et S. GIROUX. « *A smart home to assist recipes' completion (extended version)* ». Selected papers of the 2nd International Conference on Technology and Aging (ICTA). IO-Press, Accepté.

[124] H. PIGOT, D. PACHE, B. PACCOUD, S. GIROUX, J. P SAVARY, E. STIP et J. SABLIER. « MOBUS : Agenda d'aide aux déplacements ». Dans *Festival of International Conferences on Caregiving, Disability, Aging and Technology (FICCDAT), 2nd International Conference on Technology and Aging (ICTA)*, June 16-19 2007.

[125] H. PIGOT, J. P. SAVARY, J. L. METZGER, A. ROCHON et M. BEAULIEU. « Advanced technology guidelines to fulfill the needs of the cognitively impaired population ». Dans Sylvain GIROUX et Hélène PIGOT, éditeurs, *3rd International Conference On Smart Homes and health Telematic (ICOST)*, Assistive Technology Research Series, pages 25–32. IOS Press, July. 4-6, 2005 2005.

[126] M. E. POLLACK. « Assistive Technology for Aging Populations. Testimony before Special Committee on Aging - United States Senate », 2004. http ://www.eecs.umich.edu/ pollackm/Pollack-web˙files/senate-testimony.pdf.

[127] M. E. POLLACK, L. BROWN, D. COLBRY, C. E. MCCARTHY, C. OROSZ, B. PEINTNER, S. RAMAKRISHNAN et I. TSAMARDINOS. « Autominder : An intelligent cognitive orthotic system for people with memory impairment ». *Robotics and Autonomous Systems*, 44(3-4) :273–282, 2003.

[128] G. PRIVAT. « Des objets communicants à la communication ambiante ». *Les Cahiers du Numérique*, 3(4) :23–44, 2002.

[129] Y. RAHAL, P. MABILLEAU et H. PIGOT. « Bayesian Filtering and Anonymous Sensors for Localization in a Smart Home ». Dans *Proceedings of the 21st IEEE International Conference on Advanced Information Networking and Applications (AINA '07)*. IEEE, May 21-23 2007.

[130] C. REYNOLDS. « As We May Communicate ». *SIGCHI Bulletin*, 30(3) :40–44, 1998.

[131] W. A. ROGERS, B. MEYER, N. WALKER et A. D. FISK. « Functional limitations to daily living tasks in the aged : a focus group analysis ». *Human factors*, 40(1) :111–125, Mar 1998.

[132] J. ROWAN et E. D. MYNATT. « Digital Family Portrait Field Trial : Support for Aging in Place ». Dans *Conference on Human Factors in Computing Systems (CHI)*, pages 521–530, April 2-7 2005.

[133] M. N. Bouraqadi SAÂDANI. « *Un MOP Smalltalk pour l'étude de la composition et de la compatibilité des métaclasses. Application à la programmation par aspects.* ». Thèse de doctorat, Ecole Nationale Supérieure des Techniques Industrielles et des Mines de Nantes, France, 1999.

[134] M. J. SCHERER, T. HART, N. KIRSCH et M. SCHULTHESIS. « Assistive Technologies for Cognitive Disabilities ». *Critical Reviews in Physical and Rehabilitation Medicine*, 17(3) :195–215, 2005.

[135] A. SCHMIDT. « Implicit Human-Computer Interaction Through Context ». *Personal Technologies*, 4(2-3) :191–199, 2000.

[136] H. SCHULZE. « MEMOS : an interactive assistive system for prospective memory deficit compensation-architecture and functionality ». *SIGACCESS Access.Comput.*, 77-78 :79–85, 2004.

[137] A. SERNA, H. PIGOT et V. RIALLE. « Modeling the progression of Alzheimer's disease for cognitive assistance in smart homes ». *User Modeling and User-Adapted Interaction*, Accepté.

[138] N. SHADBOLT. « Ambient Intelligence ». *IEEE Intelligent Systems*, 18(4) :2–3, 2003.

[139] E. SIRIN, B. PARSIA, B. C. GRAU, A. KALYANPUR et Y. KATZ. « Pellet : A practical OWL-DL reasoner ». *Journal of Web Semantics*, 5(2), 2007.

[140] E. STIP et V. RIALLE. « Environmental Cognitive Remediation in Schizophrenia : Ethical Implications of "Smart Home" Technology ». *The Canadian Journal of Psychiatry*, 50(5) :281–291, April 2005.

[141] N. STREITZ et P. NIXON. « The Disappearing Computer ». *Communications of the ACM*, 48(3) :33–35, 2005.

[142] N. A. STREITZ, J. GEISLER, T. HOLMER, S. KONOMI, C. MÜLLER-TOMFELDE, W. REISCHL, P. REXROTH, P. Seitz et R. STEINMETZ. « i-LAND : An interactive Landscape for Creativity and Innovation ». Dans *ACM Conference on Human Factors in Computing Systems (CHI'99)*, pages 120–127, 1999.

[143] A. SZYMKOWIAK, K. MORRISON, P. GREGOR, P. SHAH, J. J. EVANS et A. WILSON. « A memory aid with remote communication using distributed technology ». *Personal Ubiquitous Comput.*, 9(1) :1–5, 2005.

[144] X. SÉRON et M. VAN DER LINDE. « *Rééducations des fonctions supérieures* », Chapitre 18, pages 183–191. Rééducation neurologique 2ème édition. Chantraine, A., 1999.

[145] E. Munguia TAPIA. « *Using Machine Learning for Real-time Activity Recognition and Estimation of Energy Expenditure* ». Thèse de doctorat, Massachusetts Institute of Technology, 2008.

[146] M. TATSUBORI, S. CHIBA, M. O KILLIJIAN et K. ITANO. « OpenJava : A Class-Based Macro System for Java ». Dans *1st OOPSLA Workshop on Reflection and Software Engineering*, LNCS 1826, pages 117–133, 2000.

[147] A. I. T. THÖNE-OTTO et K. WALTHER. « How to design an electronic memory aid for brain-injured patients : Considerations on the basis of a model of prospective memory ». *International Journal of Psychology*, 34(4) :1–9, 2003.

[148] The J. Paul Getty TRUST. « Art and Architecture Thesaurus Online ». http ://www.getty.edu/research/conducting_research/vocabularies/aat/.

[149] E. TULVING. « Multiple memory systems and consciousness ». *Human neurobiology*, 6(2) :67–80, 1987.

[150] B. ULLMER et H. ISHII. « *Emerging frameworks for tangible user interfaces* », pages 579–601. Human-Computer Interaction in the New Millenium. Carroll, J.M., 2001.

[151] A. VOINIKONIS, K. IRMSCHER et H. SCHULZE. « Distributed processing of reminding tasks within the mobile memory aid system, MEMOS ». *Personal Ubiquitous Comput.*, 9(5) :284–290, 2005.

[152] World Wide Web Consortium (W3C). « OWL Web Ontology Language Guide », February, 10 2004. http ://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[153] World Wide Web Consortium (W3C). « Extensible Markup Language (XML) 1.0 (Fourth Edition) », 2006. http ://www.w3.org/TR/REC-xml/.

[154] World Wide Web Consortium (W3C). « Namespaces in XML 1.0 (Second Edition) », 2006. http ://www.w3.org/TR/REC-xml-names/.

[155] J. S. WEBER, B. CLIPPINGDALE et M. E. POLLACK. « The Michigan Autonomous Guidance System ». Dans *Festival of International Conferences on Caregiving, Disability, Aging and Technology (FICCDAT), 2nd International Conference on Technology and Aging (ICTA)*, 2007.

[156] M. L. WEHMEYER et D. J. SANDS. *Self-Determination across the life span : Independance and choice for people with disabilities.* Paul H. Brookes, Baltimore, 1996.

[157] M. WEISER. « The computer for the 21st century ». *Scientific American*, pages 94–104, 1991.

[158] B. A. WILSON, H. C. EMSLIE, K. QUIRK et J. J. EVANS. « Reducing everyday memory and planning problems by means of a paging system : a randomised control crossover study ». *Journal of neurology, neurosurgery, and psychiatry*, 70(4) :477–482, Apr 2001.

[159] M. WU, R. BAECKER et B. RICHARDS. « Participatory design of an orientation aid for amnesics ». Dans *CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 511–520. ACM Press, 2005.

[160] M. WU, R. BAECKER et B. RICHARDS. « *Designing a Cognitive Aid for and with People who have Anterograde Amnesia* », pages 317–356. Universal Usability. Wiley & Sons, 2007.

[161] M. WU, B. RICHARDS et R. BAECKER. « Participatory design with individuals who have amnesia ». Dans *PDC 04 : Proceedings of the eighth conference on Participatory design*, pages 214–223. ACM Press, 2004.